TERMINAL SYMMETRY AS A CARRIER OF ASYMMETRIC PROCESS KNOWLEDGE: STATEWISE REFINEMENT FOR ANYTIME VERIFIED CONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many sequential construction tasks have exact terminal symmetries even though execution is directed and depends on history. Our insight is that terminal symmetry can carry independently grounded process knowledge across equivalent outcomes even when execution dynamics are asymmetric. The resulting carrier framework transports reusable structure through terminal correspondence, refines its relevance from the realized state, and certifies actions with a fixed verifier. SymBuild combines transported process and state residual ranks through ordinal rank meet. Its threshold sets are unions of the corresponding input sets, yielding a tight worst-case verifier query bound under prefix information. In computer-aided design (CAD) assembly, Mini-Programs, and exact-fill packing, SymBuild improves the area under the anytime verified success curve over Static by up to 6.77, 21.75, and 8.68 percentage points, respectively. Refresh gains extend across alternative aggregation, planning, and learned scoring methods. In Geometric Reasoning Network (GRN) target removal, direct Combined refresh achieves the lowest mean verifier query score at all three scales. It reduces distinct scored states by factors of 6.48 to 12.22 relative to refreshed population search. These results demonstrate how terminal symmetry supports knowledge reuse and statewise refinement improves verified construction.

1 INTRODUCTION

Many sequential tasks have exact terminal symmetries even though execution is directed and depends on history. A chair may be unchanged by exchanging legs, a program output by renaming, and an exact-fill packing by relabeling bins. Blockers, dependencies, and remaining capacity nevertheless determine which actions are useful during execution. Terminal equivalence offers a way to reuse process knowledge across outcomes, while accepted transitions change its decision relevance. With costly verification, exploiting this structure can focus limited effort on useful proposals. This raises a basic question: what decision information can exact terminal symmetry provide when the realizing process itself is asymmetric?

Symmetry-aware planning exploits equivalence in problems and abstractions (Fox & Long, 1999; Fox et al., 2005), while equivariant models share structure across states, actions, and trajectories (van der Pol et al., 2020; Zhao et al., 2023; Tie et al., 2025). Goal-conditioned robotics uses desired outcomes to guide execution (Seita et al., 2021; Ghosh et al., 2024). We study a complementary use of terminal information: correspondence between equivalent outcomes provides a route for transferring directed process knowledge into asymmetric execution.

Our **carrier framework** grounds directed structure in independent process evidence (Seminara et al., 2024). Theorem 1 establishes why this evidence matters: the same symmetric terminal object and observed demonstration can be consistent with different precedence and commutativity relations. Terminal correspondence transports this structure into an equivalent target frame. The realized state then updates its decision relevance after each accepted transition, and a fixed verifier certifies execution. Terminal symmetry thus carries process knowledge into sequential decisions even when execution dynamics are asymmetric.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

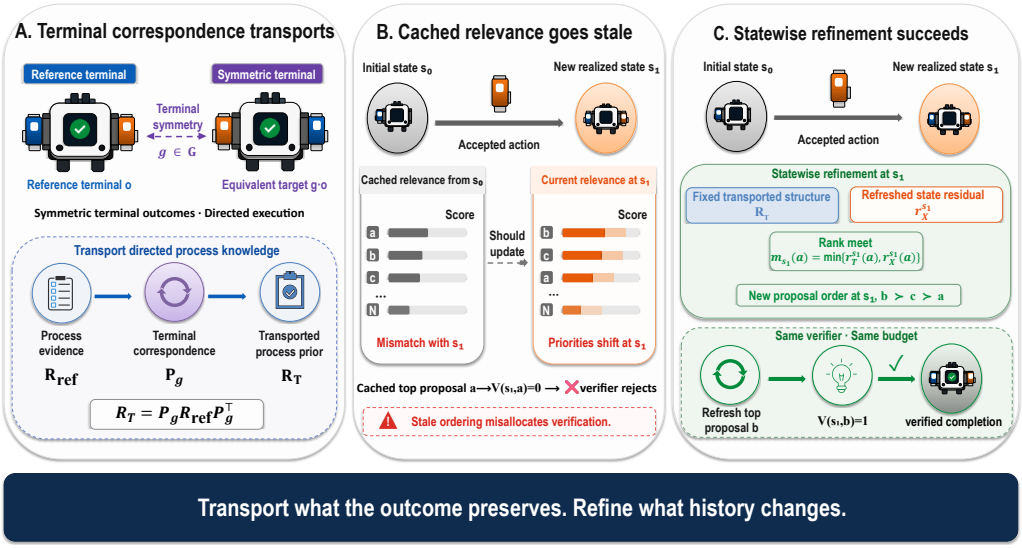

Figure 1: **Terminal correspondence transports directed process knowledge; realized state refines its decision value after accepted transitions.** (A) Process evidence grounds an asymmetric prior and a terminal automorphism transports it into the target frame. (B) After the shared first action changes feasibility, a cached composition becomes stale and the verifier rejects its next proposal. (C) SymBuild refreshes the state residual and reaches a verified plan under the same budget and verifier.

This yields a simple rule: transport what the outcome preserves and refine what history changes. A transported process prior supplies reusable order, while a state residual tracks current accessibility, dependencies, and capacity. SymBuild combines their ranks through ordinal rank meet. At each threshold, meet selects the union of the two input sets, yielding the verifier query guarantee in Theorem 2. Figure 1 shows how refresh redirects verification after a shared first action. The decomposition predicts two signatures: correspondence mismatch weakens transport value, and refresh gains emerge after accepted transitions change action relevance. SymBuild improves anytime verified success under matched budgets. Refresh gains extend across alternative aggregation and planning methods.

Our contributions are:

- **Carrier framework.** We identify terminal symmetry as a carrier of independently grounded process knowledge for asymmetric execution. The framework combines transport across equivalent outcomes with statewise refinement, turning reusable process structure into guidance for a fixed verifier.
- **SymBuild and theory.** SymBuild combines transported process and state residual ranks through ordinal rank meet. Its rank threshold sets give exact union coverage, yielding the completion guarantee of Theorem 2 and a tight worst-case verifier query bound under prefix information.
- **Evidence in multiple domains and matched evaluation.** Our protocol separates transport and refresh effects. SymBuild improves anytime verified efficiency in three construction domains, with AUC gains up to 21.75 percentage points over Static. Framework instantiations show refresh gains in four domains under alternative aggregation, planning, and learned scoring. On GRN target removal (Bouhsain et al., 2025), direct Combined refresh achieves the lowest mean verifier query score at all three scales. It reduces distinct scored states by factors of 6.48 to 12.22 relative to refreshed population search.

2 Related Work

Structure from symmetry and terminal outcomes. Equivariant RL and differentiable planning encode symmetry in state-action values, policies, operators, and action sampling (van der Pol et al.,

2020; Wang et al., 2022a; Zhao et al., 2023; Nguyen et al., 2023; Zhao et al., 2024a; Klee et al., 2026); goal-conditioned robotics separates task and motion symmetries (Mittal et al., 2024), while experience, trajectory, and generative methods reuse symmetry across trajectories, structures, and policy spaces (Lin et al., 2020; Brehmer et al., 2023; Tie et al., 2025; Zhao et al., 2024b; Kim et al., 2025; Hadjiloizou et al., 2026). Goal-conditioned robot policies map desired configurations or images to actions (Seita et al., 2021; Ghosh et al., 2024); visual planning and world model methods use intermediate plans or generated goals for long-horizon control (Ebert et al., 2018; Yang et al., 2025; Zhou et al., 2026b).

Symmetry in planning, solving, and assembly. Classical planning uses exact or approximate symmetry to reduce redundant search or guide action ordering (Fox & Long, 1999; Pochter et al., 2011; Fox et al., 2005). Dynamic SAT solving transports logical consequences under formula symmetry: symmetry propagation derives implications, while symmetric explanation learning transports explanation clauses and selects images that restrict the current partial assignment (Devriendt et al., 2012; 2017). Our framework transports process evidence into proposal information, with acceptance determined by the verifier. Assembly planning uses component symmetries and group theory to simplify geometric and kinematic reasoning (Liu & Popplestone, 1990; Popplestone et al., 1990). Across these settings, symmetry structures search, inference, or reuse at the level where equivalence is available.

Representative selection and imperfect symmetry. Canonicalization selects orbit representatives (Kaba et al., 2023; Lin & Levie, 2026; Zhou et al., 2026a), while soft, approximate, probabilistic, and partial methods modulate equivariance when symmetry is imperfect (Finzi et al., 2021; Wang et al., 2022b; Xie & Smidt, 2024; Wang et al., 2024; Lawrence et al., 2025; Chang et al., 2026). Related work also studies how symmetry interacts with evolving search states and equivariant solution families. Appendix B compares these operational roles (Devriendt et al., 2012; 2017; Lee et al., 2023; Chang et al., 2026).

Search under expensive evaluation. Partial-order planning (McAllester & Rosenblitt, 1991) and partial-order reduction (Xu et al., 2011) exploit ordering structure to constrain or reduce planning search; anytime heuristic planning and search (Likhachev et al., 2003; Hansen & Zhou, 2007; Richter & Westphal, 2010) improve solutions as computation continues, multi-heuristic search (Aine et al., 2016) combines multiple heuristics, and LazySP (Dellin & Srinivasa, 2016) targets shortest-path search with expensive edge evaluations. SayCan and VeriGraph update execution decisions from the realized state (Ichter et al., 2023; Ekpo et al., 2026). These lines motivate two complementary roles for guidance: reusable structure and state-dependent decision value. The carrier view brings them together by transporting independently grounded process structure through terminal correspondence, refining its relevance in the realized state, and leaving acceptance to a fixed verifier. Section 4.5 tests this decomposition across search interfaces.

3 Terminal Symmetry as a Carrier

3.1 Terminal Correspondence as Transport

Process evidence supplies directed process structure, and terminal symmetry provides a transport map between equivalent outcomes.

3.1.1 Construction systems and terminal lifts

A discrete construction system is $\mathcal{C} = (\mathcal{S}, \mathcal{A}, F, s_0, \Psi)$, where $F : \mathcal{S} \times \mathcal{A} \rightharpoonup \mathcal{S}$ is a deterministic partial transition and Ψ maps a final state to a semantic object. A complete executable trajectory is $\tau = (a_1, \ldots, a_T) \in \mathcal{T}_{\mathcal{C}}$ and its terminal object is $O_{\mathcal{C}}(\tau) = \Psi(s_T)$.

Let a group G act on terminal objects. A potentially partial and set-valued lift $\mathsf{L}_g(\tau)$ satisfies

$$O_{\mathcal{C}}(\tau') = g \cdot O_{\mathcal{C}}(\tau) \qquad \forall \tau' \in \mathsf{L}_g(\tau). \tag{1}$$

Equation 1 specifies terminal covariance. Symmetry is defined in this outcome model; experiments compute or estimate its correspondence from domain observations. On corresponding entity or event

indices, P_g is the selected permutation matrix. Independently obtained process evidence R_{ref} defines transported pairwise relations by

$$R_T := P_g R_{\text{ref}} P_g^\top . \tag{2}$$

Conjugation reindexes precedence, commutativity, or event correspondence. For a relation R in one frame, $R = P_g R P_g^\top$ means invariance under that permutation. Domain maps additionally transform directions or assignment labels before scoring candidate actions (Appendix D).

3.1.2 What terminal symmetry does not determine

Process evidence supplies relation content; terminal correspondence determines its indices in the target frame.

Theorem 1 (Terminal symmetry does not determine process relations)**.** *Over finite construction systems with deterministic partial transitions, no estimator that observes only a terminal object and its stabilizer can universally recover true precedence or commutativity. The result also covers one observed demonstration whose sampling policy may omit counterfactual action orders. On a witness class containing two systems, every randomized estimator has worst-case error at least* $1/2$.

A commuting diamond and a forced chain share the same terminal object, stabilizer, and demonstration, illustrating why directed relations require independent process evidence (Appendix A).

3.2 Carrier Framework and Statewise Refinement

3.2.1 Three information roles, two timescales, one verifier

Process evidence and terminal correspondence yield a transported structure R_T that is fixed for the episode; restricting it to candidates A_s induces the transported process rank $r_T^s : A_s \to [1, |A_s|]$. Evidence from the realized state supplies a state residual rank r_X^s, refreshed after accepted actions as geometry, capacity, dependencies, or extension viability change; this recomputation realizes statewise refinement. A fixed verifier $V_s(a)$ certifies execution.

The framework specifies information roles; proposal ordering remains modular. SymBuild uses ordinal rank meet so a proposal remains early whenever either rank places it early:

$$m_s(a) = \min\{r_T^s(a), r_X^s(a)\}. \tag{3}$$

Lemma 1 gives the rank threshold identity. Figure 2 and Algorithm 1 (Appendix C.1) summarize the planner. SymBuild orders actions by $(m_s(a), \kappa_s(a))$, with a deterministic secondary key, and accepts the first action approved by V_s. After acceptance, it recomputes current ranks while retaining the transported structure. Static retains initial residual information. Appendix D specifies domain rank updates and sorting keys. The same ranks can guide separate proposal queues, tested in Section 4.5 with scheduling detailed in Appendix C.6.

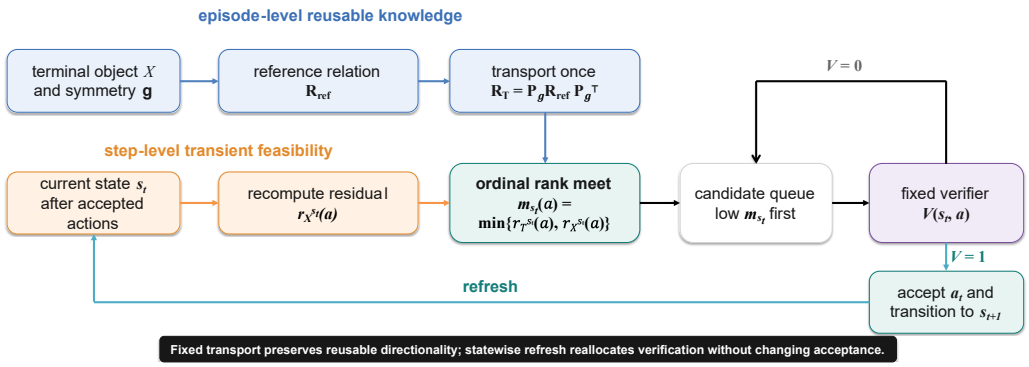

Figure 2: **Mechanism of SymBuild.** Process evidence grounds the directed prior; terminal correspondence transports it once. Process ranks use the remaining candidates; state evidence refreshes the residual after accepted transitions. Their ordinal meet orders proposals to a fixed verifier.

Instantiations. CAD transports blocker order and refreshes accessibility; Mini-Programs combine learned and observed relations and update remaining dependencies; packing transports a reference assignment and refreshes capacity-dependent viability (Appendix D). Rank meet realizes the highest rank combination rule (Ho et al., 1994) with Lemma 1's union identity. Section 4.5 tests alternative aggregation and planning methods.

3.2.2 PREFIX COVERAGE AND QUERY OPTIMALITY

For a weak rank r, its rank threshold set is $\mathrm{Top}_k(r) = \{a : r(a) \le k\}$, with size $B_r(k) = |\mathrm{Top}_k(r)|$.

Lemma 1 (Union coverage)**.** *For the rank meet in equation 3,*

$$\mathrm{Top}_k(m_s) = \mathrm{Top}_k(r_T^s) \cup \mathrm{Top}_k(r_X^s). \tag{4}$$

If either expert's rank threshold set contains an accepted action, rank meet finds one in at most $B_T^s(k) + B_X^s(k)$ verifier calls, or $2k$ for strict ranks and integer $1 \le k \le |A_s|$.

Theorem 2 (Optimality under prefix information and completion guarantee)**.** *Let $U_s(k) = \mathrm{Top}_k(r_T^s) \cup \mathrm{Top}_k(r_X^s)$. Suppose every sequence of accepted transitions from s_0 has length at most H and every candidate set is finite. Assume $V_s(a) = 1$ only if $F(s, a)$ admits a complete suffix, and every incomplete reachable state contains an action $a \in U_s(k_s)$ with $V_s(a) = 1$. With sufficient query budget,* SymBuild *completes in $T \le H$ accepted transitions using at most*

$$\sum_{t=0}^{T-1} |U_{s_t}(k_{s_t})| \le \sum_{t=0}^{T-1} \big(B_T^{s_t}(k_{s_t}) + B_X^{s_t}(k_{s_t})\big) \tag{5}$$

queries. At one state, meet attains the tight worst-case bound $|U_s(k)|$ among deterministic query policies when the only promise is a feasible action in $U_s(k)$.

Union coverage supplies an accepted action, and extension preservation allows this argument to continue to completion. Appendix A.3 proves the guarantees and details finite-budget execution and completeness over the full candidate graph.

3.3 ANYTIME VALUE UNDER EXPENSIVE VERIFICATION

Statewise refinement reallocates expensive verification, so performance is naturally measured by completion before an uncertain resource limit. Let Q_π be the verifier resource required to reach a complete verified solution and $F_\pi(b) = \Pr(Q_\pi \le b)$. For a predeclared budget distribution μ,

$$\mathrm{AUC}_\mu(\pi) = \sum_{b \in \mathcal{B}} \mu(b) F_\pi(b) = \Pr_{\omega, B}(Q_\pi(\omega) \le B), \qquad B \sim \mu. \tag{6}$$

Thus anytime AUC is expected success under random interruption. Appendix A gives the formal derivation and the capped cost identity.

A shared first action can leave Static testing an infeasible proposal, while refresh selects a feasible one and completes within two calls using the same prior and fixed verifier. Proposition 1 in Appendix A.5 gives the construction and proof. Figure 4 tests this pattern empirically.

4 EXPERIMENTS

Experiments evaluate the carrier framework in four domains. SymBuild receives performance comparisons, ablations, and mechanism analyses in CAD assembly, Mini-Programs, and exact-fill packing, with its ordinal adaptation evaluated in target removal. Additional framework tests use alternative aggregation in the construction domains, planning variants in Mini-Programs, and learned GRN scoring with direct ranking and population-based search.

4.1 DOMAINS AND PROTOCOL

We evaluate CAD assembly (72 assemblies in the call panel and 48 in the independent work panel), 480 Mini-Programs, 480 symmetric exact-fill packings, and 1,135 solvable target removal episodes

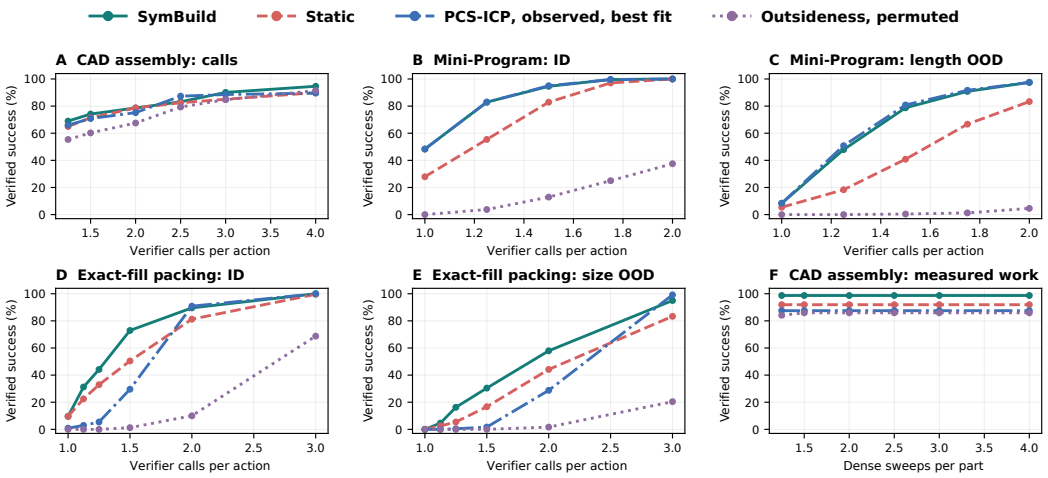

Figure 3: **Statewise refinement shifts the frontier of verified success throughout the predeclared budget range.** Other single-channel and permuted controls appear where defined; the flat CAD work curves show that the gain comes from additional verified completions.

from official GRN out-of-distribution (OOD) scenes. ID denotes in-distribution settings and pp denotes percentage points. Static omits post-transition proposal refresh while accepted actions update candidates and required planner state. Matched pairs share instances, observables, candidates, verifier, and budgets. Primary construction pairs and their post-transition diagnostics also share first actions. GRN and CDGS use five seeds, one frozen GRN checkpoint, and a $5N$ cap. Direct ranking pairs also share first queried actions. Appendix D specifies domain-specific seeds, construction, and inference. Table 7 summarizes the three SymBuild domains. Appendix C.6 gives the planning controls, and Appendix C.7 adapts Compositional Diffusion with Guided Search (CDGS, Mishra et al., 2026). Appendix Table 21 identifies the related methods and their roles.

4.2 SymBuild in Three Construction Domains

In CAD assembly, Mini-Programs, and exact-fill packing, SymBuild improves verified efficiency over Static in all six aggregates (Table 1). With transported process structure and verifier fixed, SymBuild refreshes state relevance after accepted transitions, while Static retains its initial relevance ranking. Figure 3 also includes PCS-ICP, a CAD control using trimmed iterative closest point (ICP) matching (Besl & McKay, 1992; Chetverikov et al., 2002) in the process coordinate system (PCS).

The same ordering holds across the predeclared budgets, with separations at intermediate budgets in CAD calls, Mini-Programs, and packing (Figure 3). CAD work has reached its plateau. Appendix C.4 examines smaller budgets. Pointwise ordering preserves the result under nonnegative budget weighting, with strict gains whenever separating budgets receive positive mass (Appendix A). Appendix Table 12 gives exact values. Appendix Table 10 reports subgroup replications.

Table 1: **SymBuild in three construction domains.** Matched pairs share channels, candidates, first actions, and verifier. Static omits post-transition refresh. AUC entries are percentages with paired 95% confidence intervals (CIs). Cost reduction is the paired decrease in capped normalized verifier cost, with unsuccessful runs charged the cap.

Domain	Setting	N	Static	SymBuild	ΔAUC [95% CI]	Cost reduction
CAD	ID (calls)	72	78.73	**81.60**	+2.86 [.12, 5.76]	.0253
CAD	ID (work)	48	91.93	**98.70**	+6.77 [1.30, 14.06]	.0523
Program	ID (calls)	240	72.67	**85.08**	+12.42 [9.50, 15.33]	.0737
Program	length OOD	240	42.92	**64.67**	+21.75 [18.33, 24.92]	.1351
Packing	ID (calls)	240	49.38	**57.92**	+8.54 [7.01, 10.14]	.0539
Packing	size OOD	240	25.35	**34.03**	+8.68 [7.01, 10.42]	.0981

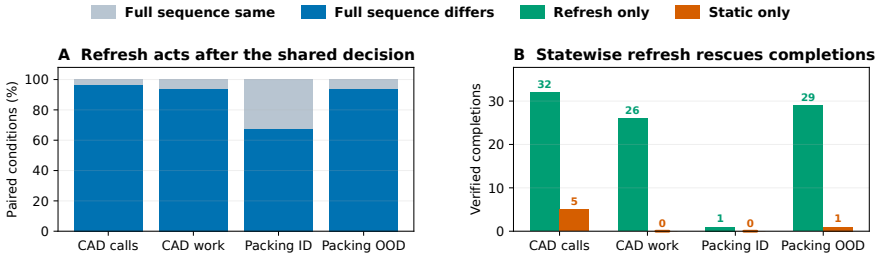

Figure 4: **Statewise refresh acts after the shared first decision.** Accepted sequences diverge after the transition, and verified completions unique to the refreshed policy dominate.

4.3 Post-Transition Effect of Statewise Refinement

Proposition 1 predicts that refreshed and static policies share the first accepted action and diverge after the realized transition changes residual relevance. The primary rank meet comparison shows this pattern directly: SymBuild and Static agree on the first accepted action in the paired CAD and packing panels, then diverge, with more verified completions unique to SymBuild (Figure 4). An independent CAD panel reproduces the post-transition effect (Table 2(b)).

4.4 Information Roles and Robustness in SymBuild

The ablations separate the two information roles. CAD assembly and exact-fill packing are more sensitive to transport; state evidence dominates in Mini-Programs. In the independent CAD panels, Algorithm compares SymBuild with PCS-ICP and an outsideness control adapted from ASAP (Tian et al., 2024), Timing compares SymBuild with Static, and Alignment compares aligned with permuted transport; all share the verifier. Observed process topology adds complementary mechanism evidence (Table 2). Here, IKEA Manuals at Work (Liu et al., 2024) provides real furniture-assembly videos aligned with parts, manuals, and substeps. Appendix D gives the protocol and definitions. Transport contributes most when correspondence relocates physical constraints; state evidence can dominate when the residual exposes current precedence. Complete channel comparisons are in Appendix C.1.

Table 2: **Evidence for distinct information roles.** (a) AUC (%) ablations; (b) CAD mechanism point estimates. In (b), Algorithm reports two comparisons, first with PCS-ICP and then outsideness. Timing lists final success for SymBuild then Static. Alignment lists aligned then permuted transport. In each comparison, success gain is the first condition minus the comparator, and call saving is the comparator minus the first condition. Positive values therefore favor the first condition. (c) Gains in Top-1 accuracy for predicting the observed next action (pp), with paired 95% CIs.

(a) Refresh and information channel ablations (AUC ↑)

Domain	Panel	N	SymBuild	Static	Process-only	State-only
CAD	calls	72	81.60	78.73	79.57	73.12
CAD	work	48	98.70	91.93	87.50	85.63
Program	ID	240	85.08	72.67	84.83	85.17
Program	length OOD	240	64.67	42.92	63.58	65.83
Packing	ID	240	57.92	49.38	50.21	38.26
Packing	size OOD	240	34.03	25.35	28.40	21.67

(b) Independent CAD mechanism contrasts

Test	Panel	Final success (% ↑)	Comparator	Success gain (pp ↑)	Call saving (calls per part ↑)
Algorithm	120 objects × 4 lifts	84.58	PCS-ICP	+5.00	+.143
			outsideness	+7.71	+.168
Timing	120 objects × 4 lifts	84.38 / 80.00	Static	+4.38	+.152
Alignment	60 objects × 4 lifts	81.25 / 65.00	permuted transport	+16.25	+.637

(c) Evidence from observed process topology (Top-1 accuracy gain ↑)

Evidence	Panel	Process structure	Action core	Hybrid
IKEA topology	36 families, 97 videos, 1,022 states	+14.14 [6.41, 20.18]	+19.20 [10.26, 25.56]	+2.64 [.77, 4.38]

Beyond the tabulated contrasts, the Timing panel shows mean verifier calls per part of 1.569 for SymBuild and 1.721 for Static, with 96.46% sequence disagreement. In (c), Process structure uses states with terminal ambiguity, Action core restricts to cases where the realized prefix breaks every

relevant transport, and Hybrid uses the section model only on ambiguous states. Appendix Table 9 gives the CAD CIs.

Correspondence and certification tests show two robustness effects. Transport value decreases as correspondence mismatch increases, while SymBuild retains positive true AUC gains over Static when certification is approximate (Table 3).

Table 3: **Correspondence fidelity and practical verification value.** (a) Each cell gives AUC first and final verified success second, both in percent, under graded deterministic mismatch of action identities; Appendix C.3 gives construction details. (b) At 5% verifier noise, ΔAUC is SymBuild minus Static; FC denotes false certification, meaning completion without true success.

(a) Correspondence fidelity controls transport value (AUC and success ↑)					
Requested mismatch	0%	10%	25%	50%	100%
Program ID	**85.08/100.00**	75.33/98.33	63.42/95.42	40.58/77.92	13.58/36.67
Program length OOD	**64.67/97.50**	49.58/88.75	32.50/70.00	11.42/31.67	.75/2.92

(b) Approximate verification preserves refresh value			
Condition	ID ΔAUC ↑	Length OOD ΔAUC ↑	Reliability marker
5% false positive	+17.47 [13.70, 21.25]	+20.48 [17.18, 23.76]	FC reduction ↑: 10.89 pp ID; 14.00 pp OOD
5% false negative	+8.65 [6.25, 11.13]	+15.01 [12.36, 17.75]	No false certification on either split

These interventions trace both parts of the information flow: transport value falls with correspondence mismatch, while refresh gains persist throughout the 1–10% noise sweep and false certification is lower under false positive noise (Appendix Table 11).

4.5 Framework Tests Beyond SymBuild: Aggregation and Planning

The Mini-Program planning controls keep the 480 objects, candidates, verifier, and budgets fixed. Table 4 compares alternative planners and tests refresh within different planning mechanisms. SymBuild outperforms the canonical slice and weighted A* with LazySP-style replanning on both splits, demonstrating more effective proposal ordering from shared process and state information under limited verification budgets.

Table 4: **Comparison under alternative planning methods.** AUC gaps with paired 95% CIs. Appendix Table 16 adds final success, selector time, and search expansions.

Comparison	ID gap ↑	Length OOD gap ↑	Interface feature
Difference between SymBuild and canonical slice with static meet	+12.58 [9.67, 15.50]	+21.75 [18.42, 25.17]	canonical representative
Difference between SymBuild and weighted A* with LazySP-style replanning	+4.42 [2.58, 6.33]	+4.25 [1.83, 6.67]	lazy replanning
Difference between POCL-style refresh and fixed	+1.08 [.25, 2.00]	+3.83 [2.58, 5.08]	partial-order agenda
Difference between MHA*-style refresh and fixed	−.42 [−1.75, .83]	+5.00 [2.83, 7.17]	scheduling with two queues

Within partial-order causal-link (POCL)-style (McAllester & Rosenblitt, 1991) and MHA*-style (Aine et al., 2016) planning, refresh raises length OOD AUC by 3.83 and 5.00 pp, respectively. These paired gains extend the framework to partial-order agendas and scheduling with two queues. Appendix Table 16 reports completion and computational costs. Alternative aggregation provides complementary evidence: the SayCanPay-adapted product rule (Hazra et al., 2024) retains refresh gains in all six construction panels after the shared first action, while a weighted sum selected on development data strengthens Mini-Program results (Appendix Tables 15 and 14). Rank meet is the SymBuild choice because it outperforms Borda count (Ho et al., 1994) and rank join on the CAD development panel and supplies the rank threshold identity in Lemma 1 (Appendix C.5).

4.6 Learned Scoring and CDGS-style Search in GRN Target Removal

Following transport and refinement tests in the construction domains, GRN target removal evaluates SymBuild-style ordinal fusion and refresh with identity correspondence on scene objects. Combined sums learned GRN feasibility and annotated process scores. Rank meet fuses their ordinal ranks. Direct ranking selects individual actions, while CDGS-style search evaluates short imagined sequences. Static caches initial raw scores, and refresh recomputes them after accepted removals. Appendix C.7 gives the full paired protocol.

Table 5: **GRN scoring and population-based search inspired by CDGS.** Panels (a) and (b) test refresh against matched Static controls using mean verifier query scores (↓). Failures receive $5N + 1$ under a $5N$ cap. Each Δ is refresh minus Static for the same method with a paired 95% CI. Negative values favor refresh. In (a), bold and underlined means mark the best and second-best scores per scale. Panel (b) repeats direct Combined refresh for comparison. Panel (c) reports distinct scored GRN states and sampled sequence positions per target. The reduction factor divides CDGS–refresh state counts by direct Combined refresh state counts.

(a) Direct action ranking

Split	Targets	Method	Static mean	Refresh mean	Δ [95% CI]
OOD10	129	GRN	5.457	5.132	-0.326 $[-0.743, 0.066]$
OOD10	129	Rank meet	5.352	4.747	-0.605 $[-1.433, -0.025]$
OOD10	129	Combined	<u>3.349</u>	**2.597**	-0.752 $[-1.279, -0.270]$
OOD15	432	GRN	8.213	6.734	-1.479 $[-2.053, -0.978]$
OOD15	432	Rank meet	7.097	6.127	-0.970 $[-1.472, -0.560]$
OOD15	432	Combined	<u>4.523</u>	**3.349**	-1.174 $[-1.500, -0.865]$
OOD20	574	GRN	11.027	8.028	-2.999 $[-4.372, -1.879]$
OOD20	574	Rank meet	9.319	7.608	-1.711 $[-2.608, -0.995]$
OOD20	574	Combined	<u>6.332</u>	**3.873**	-2.459 $[-3.225, -1.774]$

(b) Population-based search

Split	Targets	CDGS–GRN	CDGS–static	CDGS–refresh	Combined refresh (direct ranking)	Δ CDGS [95% CI]
OOD10	129	5.447	4.184	<u>3.047</u>	**2.597**	-1.138 $[-2.145, -.341]$
OOD15	432	8.515	5.844	<u>3.828</u>	**3.349**	-2.016 $[-2.612, -1.463]$
OOD20	574	8.482	7.489	<u>4.158</u>	**3.873**	-3.331 $[-4.224, -2.493]$

(c) Scored states and sampled sequence positions per target

Split	Targets	Combined refresh GRN states	CDGS–refresh GRN states	CDGS–refresh scored positions	CDGS–GRN scored positions	State count reduction factor
OOD10	129	**2.00**	<u>12.96</u>	144.85	353.33	6.48
OOD15	432	**2.64**	<u>23.64</u>	214.67	624.21	8.96
OOD20	574	**3.14**	<u>38.32</u>	258.60	669.05	12.22

Across all three scales, Combined, rank meet, and CDGS-style search reduce mean query scores against matched Static controls, with paired CIs below zero (Table 5). SymBuild-style rank meet lowers mean scores by 11.30–18.36% and the OOD20 95th-percentile score from 25 to 17. Tail scores also fall on OOD10 and OOD15 (Appendix Table 18). Rank meet refresh also lowers mean query scores in CDGS-style search by 12.48–19.10% across scales, with paired gains and scoring costs in Appendix Tables 19 and 20. Direct Combined refresh achieves the lowest means with aggregation suited to domain signals and requires 6.48–12.22 times fewer distinct scored states than CDGS–refresh. Including initial scoring, its query savings offset additional scoring cost above verifier latencies of 12.55–31.44 ms per query (Appendix Tables 23 and 24).

5 Discussion and Limitations

Terminal correspondence reuses directed process knowledge across equivalent outcomes, while statewise refinement adapts its relevance to execution history. Results across aggregation, planning, and learned scoring support domain-specific combinations of process and state signals, exemplified by Combined's query efficiency in GRN target removal. The framework is especially useful when reliable correspondence and reusable process evidence guide decisions with costly verification. Correspondence fidelity governs the quality of transported information, and verifier reliability governs execution certification (Appendix C.3). Under Theorem 2's assumptions and sufficient budget, rank meet guarantees completion with a tight worst-case query bound through union coverage and suffix preservation (Appendix A.3). Future work will explore aggregation adapted to domain signals and realized states, and joint reasoning over multiple terminal correspondences.

6 Conclusion

Terminal correspondence transports process directionality; realized state updates its relevance. SymBuild makes this separation analyzable and improves verified efficiency in three construction domains, with its ordinal adaptation also improving target removal. Framework tests cover four domains, with refresh gains under alternative aggregation, planning, and learned scoring. In short, transport what the outcome preserves, refine what history changes, and leave acceptance to the verifier.

REPRODUCIBILITY STATEMENT

Appendices C and D provide the experimental protocols, controls, results, and computing environment. The submitted anonymous code implements the core algorithmic components, including rank-based fusion, verified planning with state refresh after accepted transitions, and the matched Static control. The artifact also includes fixed experimental configurations with seeds, budgets, model and verifier settings, an executable construction for Proposition 1, and unit tests for ranking and refresh semantics. The complete experimental codebase will be released as open source upon acceptance.

AI USE STATEMENT

An LLM was used to assist with writing and experimental implementation. For writing, the LLM was used to revise and polish text based on descriptions already written by the authors, with the aim of improving clarity and precision. In addition, for the experimental details reported in Appendix D, after the authors had completed the initial descriptions, the LLM was used to assist in comparing these descriptions with the actual experimental code and in reviewing, revising, and polishing the corresponding text. The aim was to improve readability and the accuracy and rigor of the reported experimental details, thereby improving the fairness and completeness of the presentation of the experimental protocol. The LLM was also used to assist in checking whether our coverage of related work was comprehensive, and with reference to the original texts of the related work, to polish Appendix B on the basis of the authors' draft.

For the experiments, we first developed the experimental framework, design, and initial settings. The LLM was then used to provide feedback on the proposed framework, including identifying possible omissions, assessing consistency across experimental conditions, and examining the completeness and fairness of the comparisons. Based on this feedback, we revised and improved the experimental framework. The LLM was subsequently used to assist with implementing the experimental code under the revised framework and to generate the random seed values used in the experiments. All random seed values generated for this purpose were used in the experiments and are reported in Appendices.

Scientific discovery, formulation of the research problem, formulation of mathematical claims, interpretation of experimental results, and determination of scientific conclusions were carried out independently by the authors. We reviewed all revisions, feedback, code, and configurations produced with the assistance of the LLM and tested the code where applicable. All final decisions were made by the authors. We take full responsibility for the manuscript, experimental design, experimental implementation, and code.

REFERENCES

Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic A*. *The International Journal of Robotics Research*, 35(1-3):224–243, 2016. doi: 10.1177/0278364915594029.

Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

Smail Ait Bouhsain, Rachid Alami, and Thierry Simeon. Learning geometric reasoning networks for robot task and motion planning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Johann Brehmer, Joey Bose, Pim de Haan, and Taco Cohen. EDGI: Equivariant diffusion for planning with embodied agents. In *Advances in Neural Information Processing Systems*, 2023.

Junwoo Chang, Minwoo Park, Joohwan Seo, Roberto Horowitz, Jongmin Lee, and Jongeun Choi. Partially equivariant reinforcement learning in symmetry-breaking environments. In *International Conference on Learning Representations*, 2026.

D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *2002 International Conference on Pattern Recognition*, volume 3, pp. 545–548 vol.3, 2002. doi: 10.1109/ICPR.2002.1047997.

Christopher M. Dellin and Siddhartha S. Srinivasa. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2016.

Jo Devriendt, Bart Bogaerts, Broes De Cat, Marc Denecker, and Christopher Mears. Symmetry propagation: Improved dynamic symmetry breaking in SAT. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, pp. 49–56, 2012. doi: 10.1109/ICTAI.2012.16.

Jo Devriendt, Bart Bogaerts, and Maurice Bruynooghe. Symmetric explanation learning: Effective dynamic symmetry handling for SAT. In *Theory and Applications of Satisfiability Testing – SAT 2017*, volume 10491 of *Lecture Notes in Computer Science*, pp. 83–100. Springer, 2017. doi: 10.1007/978-3-319-66263-3_6.

Frederik Ebert, Sudeep Dasari, Alex X. Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In *Proceedings of the 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 983–993, 2018.

Daniel Ekpo, Mara Levy, Saksham Suri, Chuong Huynh, Archana Swaminathan, and Abhinav Shrivastava. VeriGraph: Scene graphs for execution verifiable robot planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Marc Anton Finzi, Gregory Benton, and Andrew Gordon Wilson. Residual pathway priors for soft equivariance constraints. In *Advances in Neural Information Processing Systems*, 2021.

Maria Fox and Derek Long. The detection and exploitation of symmetry in planning problems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 956–961, 1999.

Maria Fox, Derek Long, and Julie Porteous. Abstraction-based action ordering in planning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090.

Loizos Hadjiloizou, Rodrigo Pérez-Dattari, and Noémie Jaquier. Symmetries here and there, combined everywhere: Cross-space symmetry compositions in robotics, 2026.

Eric A. Hansen and Rong Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 2007.

Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. SayCanPay: Heuristic planning with large language models using learnable domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Tin Kam Ho, J.J. Hull, and S.N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994. doi: 10.1109/34.273716.

Brian Ichter et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15546–15566. PMLR, 23–29 Jul 2023.

Hohyun Kim, Seunggeun Lee, and Min-Hwan Oh. Symmetry-aware GFlowNets. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

David M. Klee, Boce Hu, Andrew Cole, Heng Tian, Dian Wang, Robert Platt, and Robin Walters. RAVEN: End-to-end equivariant robot learning with RGB cameras. In *International Conference on Learning Representations*, 2026.

Hannah Lawrence, Vasco Portilheiro, Yan Zhang, and Sékou-Oumar Kaba. Improving equivariant networks with probabilistic symmetry breaking. In *International Conference on Learning Representations*, 2025.

Byeongho Lee, Yonghyeon Lee, Seungyeon Kim, MinJun Son, and Frank C. Park. Equivariant motion manifold primitives. In *7th Annual Conference on Robot Learning*, 2023.

Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, 2003.

Ya-Wei Eileen Lin and Ron Levie. Adaptive canonicalization with application to invariant anisotropic geometric networks. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yijiong Lin, Jiancong Huang, Matthieu Zimmer, Yisheng Guan, Juan Rojas, and Paul Weng. Invariant transform experience replay: Data augmentation for deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6615–6622, 2020. doi: 10.1109/LRA.2020.3013937.

Yanxi Liu and Robin J. Popplestone. Symmetry constraint inference in assembly planning: Automatic assembly configuration specification. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 1038–1044, 1990.

Yunong Liu, Cristobal Eyzaguirre, Manling Li, Shubh Khanna, Juan Carlos Niebles, Vineeth Ravi, Saumitra Mishra, Weiyu Liu, and Jiajun Wu. IKEA manuals at work: 4d grounding of assembly instructions on internet videos. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Sharath Matada, Luke Bhan, Yuanyuan Shi, and Nikolay Atanasov. Generalizable motion planning via operator learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991.

Utkarsh Aashu Mishra, David He, Yongxin Chen, and Danfei Xu. Compositional diffusion with guided search for long-horizon planning. In *International Conference on Learning Representations*, 2026.

Mayank Mittal, Nikita Rudin, Victor Klemm, Arthur Allshire, and Marco Hutter. Symmetry considerations for learning task symmetric robot policies. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7433–7439, 2024. doi: 10.1109/ICRA57147.2024.10611493.

Hai Huu Nguyen, Andrea Baisero, David Klee, Dian Wang, Robert Platt, and Christopher Amato. Equivariant reinforcement learning under partial observability. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3309–3320. PMLR, 06–09 Nov 2023.

Nir Pochter, Aviv Zohar, and Jeffrey S. Rosenschein. Exploiting problem symmetries in state-based planners. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Robin J. Popplestone, Yanxi Liu, and Rich Weiss. A group theoretic approach to assembly planning. *AI Magazine*, 11(1):82–97, 1990.

Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 2010.

Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4568–4575, 2021. doi: 10.1109/ICRA48506.2021.9561391.

Luigi Seminara, Giovanni Maria Farinella, and Antonino Furnari. Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Yunsheng Tian, Karl D. D. Willis, Bassel Al Omari, Jieliang Luo, Pingchuan Ma, Yichen Li, Farhad Javid, Edward Gu, Joshua Jacob, Shinjiro Sueda, Hui Li, Sachin Chitta, and Wojciech Matusik. ASAP: Automated sequence planning for complex robotic assembly with physical feasibility. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4380–4386, 2024. doi: 10.1109/ICRA57147.2024.10611595.

Chenrui Tie, Yue Chen, Ruihai Wu, Boxuan Dong, Zeyi Li, Chongkai Gao, and Hao Dong. ET-SEED: Efficient trajectory-level SE(3) equivariant diffusion policy. In *The Thirteenth International Conference on Learning Representations*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Elise van der Pol, Daniel E. Worrall, Herke van Hoof, Frans A. Oliehoek, and Max Welling. MDP homomorphic networks: Group symmetries in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Dian Wang, Robin Walters, Xupeng Zhu, and Robert Platt. Equivariant q learning in spatial action spaces. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1713–1723. PMLR, 08–11 Nov 2022a.

Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23078–23091. PMLR, 17–23 Jul 2022b.

Rui Wang, Elyssa Hofgard, Han Gao, Robin Walters, and Tess Smidt. Discovering symmetry breaking in physical systems with relaxed group convolution. In *Forty-first International Conference on Machine Learning*, 2024.

YuQing Xie and Tess Smidt. Equivariant symmetry breaking sets. *Transactions on Machine Learning Research*, 2024.

You Xu, Yixin Chen, Qiang Lu, and Ruoyun Huang. Theory and algorithms for partial order based reduction in planning, 2011.

Wenyan Yang, Ahmet Tikna, Yi Zhao, Yuying Zhang, Luigi Palopoli, Marco Roveri, and Joni Pajarinen. Extracting visual plans from unlabeled videos via symbolic guidance. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3995–4018, 2025.

Linfeng Zhao, Xupeng Zhu, Lingzhi Kong, Robin Walters, and Lawson L.S. Wong. Integrating symmetry into differentiable planning with steerable convolutions. In *The Eleventh International Conference on Learning Representations*, 2023.

Linfeng Zhao, Owen Howell, Xupeng Zhu, Jung Yeon Park, Zhewen Zhang, Robin Walters, and Lawson L. S. Wong. Equivariant action sampling for reinforcement learning and planning. In *The 16th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2024a.

Lingxiao Zhao, Xueying Ding, and Leman Akoglu. PARD: Permutation-invariant autoregressive diffusion for graph generation. In *Advances in Neural Information Processing Systems*, 2024b.

Cai Zhou, Zijie Chen, Zian Li, Jike Wang, Kaiyi Jiang, Pan Li, Rose Yu, Muhan Zhang, Stephen Bates, and Tommi Jaakkola. Rethinking diffusion models with symmetries through canonicalization with applications to molecular graph generation, 2026a.

Pengfei Zhou, Liliang Chen, Shengcong Chen, Di Chen, Wenzhi Zhao, Rongjun Jin, Guanghui Ren, and Jianlan Luo. Act2Goal: From World Model To General Goal-conditioned Policy. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026b. doi: 10.15607/RSS.2026.XXII.015.

APPENDICES

A Proofs **14**

A.1 Proof of Theorem 1 14

A.2 Proof of Lemma 1 15

A.3 Proof of Theorem 2 15

A.4 Random interruption identity 15

A.5 Proof of Proposition 1 15

A.6 Capped cost identity 16

B Operational Relation to Prior Symmetry Uses **16**

C Extended Empirical Evidence **17**

C.1 SymBuild Baselines and Ablations in Three Construction Domains 17

C.2 SymBuild Replication by Group 19

C.3 Correspondence Fidelity and Approximate Verification 19

C.3.1 Approximate Verification 20

C.4 Complete Anytime Frontiers 20

C.5 Alternative Aggregation Beyond SymBuild 21

C.6 Alternative Planning Methods 22

C.7 GRN and Population-Based Search 23

C.7.1 Wall-Clock Verification Value 28

D Experimental Protocols and Environment **29**

D.1 Shared Evaluation and Statistics 30

D.2 CAD Assembly 30

D.3 Mini-Programs 33

D.4 Exact-Fill Packing 34

D.5 IKEA Observed Process Topology 35

D.6 Alternative Aggregation and Matched Controls 35

D.7 Implementation Sources and Environment 36

A PROOFS

A.1 PROOF OF THEOREM 1

Let both systems use the action set $\{a, b\}$ and the same terminal object o with a nonidentity automorphism g that exchanges two indistinguishable terminal roles, so $g \cdot o = o$ and the terminal stabilizers coincide. In $\mathcal{C}_{\|}$, define the diamond transitions $s_0 \xrightarrow{a} s_a \xrightarrow{b} s_T$ and $s_0 \xrightarrow{b} s_b \xrightarrow{a} s_T$; then a and b commute at s_0. In $\mathcal{C}_{\to}$, define only $q_0 \xrightarrow{a} q_a \xrightarrow{b} q_T$, so $a \prec b$ is forced. Thus the same nontrivial terminal symmetry is compatible with different process relations. Let the behavior policy emit ab with probability one in both systems. The observation distributions are therefore identical whether the estimator sees only the terminal object or also one demonstration.

Fix the shared observation z. If a randomized estimator outputs the first relation with probability p and the second with probability q, then $p + q \leq 1$. Its errors on the two systems are at least $1 - p$ and $1 - q$, hence

$$\max\{1 - p, 1 - q\} \geq 1 - \frac{p + q}{2} \geq \frac{1}{2}. \tag{7}$$

Thus no universally consistent estimator exists. Repeated terminal observations preserve the same point-mass observation distribution.

A.2 Proof of Lemma 1

For every action a,

$$m_s(a) \leq k \iff \min\{r_T^s(a), r_X^s(a)\} \leq k \iff r_T^s(a) \leq k \text{ or } r_X^s(a) \leq k. \tag{8}$$

This proves the set identity. Its cardinality is at most $B_T^s(k) + B_X^s(k)$. If a feasible action is in the union, an ordering that exhausts meet ranks through k must encounter one within that many queries. For strict ranks and integer $1 \leq k \leq |A_s|$, $B_T = B_X = k$.

A.3 Proof of Theorem 2

At each incomplete state, Lemma 1 places every member of $U_s(k_s)$ before every action outside it. Coverage supplies an accepted member within $|U_s(k_s)|$ calls. Extension preservation leaves a complete suffix after the transition. The horizon bound therefore gives completion in $T \leq H$ accepted transitions. Summing the union sizes over the visited states $s_0, \ldots, s_{T-1}$ proves the verifier-call bound. A finite cap returns the current trajectory when calls are exhausted. With $k_s = |A_s|$, $U_s(k_s) = A_s$, giving relative completeness.

For optimality, fix a state and any deterministic query policy. An adversary returns infeasible for every action outside $U_s(k)$ and for queried union members until its last unqueried member, which is feasible. Repeated queries retain their previous labels. This satisfies the coverage promise and forces at least $|U_s(k)|$ calls; a policy that never exhausts the union cannot certify success. Meet queries the union before outside actions and matches the lower bound.

Verifier safety and covariance. Acceptance is determined by the fixed verifier for every proposal order. If the selected correspondence induces a candidate bijection $\varphi_g : A_s \to A_{gs}$ and both ranks are covariant, with $r_T^{gs}(\varphi_g(a)) = r_T^s(a)$ and $r_X^{gs}(\varphi_g(a)) = r_X^s(a)$, then $m_{gs}(\varphi_g(a)) = m_s(a)$. Here φ_g includes the domain's entity and action-parameter maps. Equality among orbit actions within one realized state additionally requires stabilizer invariance; covariance concerns transformed states.

A.4 Random interruption identity

Conditioning on the independently drawn budget gives

$$\Pr(Q_\pi \leq B) = \sum_{b \in \mathcal{B}} \Pr(B = b) \Pr(Q_\pi \leq b) = \sum_{b \in \mathcal{B}} \mu(b) F_\pi(b). \tag{9}$$

Pointwise cumulative distribution function (CDF) dominance implies dominance for every nonnegative weighting. Conversely, choosing a point mass at each b recovers each pointwise inequality.

A.5 Proof of Proposition 1

Proposition 1 (Refresh versus static separation)**.** *There exists a problem with two nonterminal decision states in which refreshed and static policies use the same terminal prior, fixed verifier, and first action, yet refresh completes in two calls while static is interrupted at budget two.*

Let s_0 have candidates p, a, b and accept only p, leading to s_1. At s_1, a is rejected and b is accepted, completing the task. Each query costs one call. The ranks and their meet are

State	Action	r_T	r_X	$\min(r_T, r_X)$
s_0	p	1	1	1
s_0	a	2.5	2	2
s_0	b	2.5	3	2.5
s_1	a	1.5	2	1.5
s_1	b	1.5	1	1

The fixed process scores tie a, b; restricting them to remaining candidates changes their average rank from 2.5 to 1.5. Static caches the initial order p, a, b and retains a, b after accepting p. Refresh orders b, a at s_1. Budget two therefore gives the traces p, a for Static and p, b for refresh. Both decisive orders follow from the primary meet ranks.

A.6 Capped cost identity

For dense integer budgets, anytime completion is also linked to capped time to solution. For a nonnegative integer q and cap M, $\min\{q, M\} = \sum_{b=0}^{M-1} \mathbb{I}\{q > b\}$. Taking expectations and using $\Pr(Q > b) = 1 - F(b)$ gives

$$1 - \frac{\mathbb{E}[\min\{Q_\pi, M\}]}{M} = \frac{1}{M} \sum_{b=0}^{M-1} F_\pi(b). \tag{10}$$

B Operational Relation to Prior Symmetry Uses

Table 6 compares the symmetry locus and information role of the closest historical and modern precedents. Classical planning and SAT expose complementary forms of reuse. Fox et al. (2005) use almost symmetry derived from abstraction to guide forward action ordering. In SAT, known automorphisms preserve logical consequences: symmetry propagation derives implications, and symmetric explanation learning maps explanation clauses to consequences useful under the current partial assignment (Devriendt et al., 2012; 2017). Our framework transports independently identified process evidence through terminal correspondence. The realized state refines this proposal information, and the fixed verifier determines action acceptance.

The broader landscape includes MDP homomorphic networks and equivariant Q learning methods that encode joint state-action symmetry in value and policy structure (van der Pol et al., 2020; Wang et al., 2022a), soft or approximate equivariance under misspecified symmetries (Finzi et al., 2021; Wang et al., 2022b), and graph generation methods that incorporate symmetry (Zhao et al., 2024b; Kim et al., 2025). Across these settings, the carrier framework separates what is supplied by process evidence, what terminal correspondence can carry across equivalent outcomes, and what must be updated from the realized state.

Table 6: **Operational comparison of symmetry uses.** The key distinction is where symmetry is assumed and what information it contributes to decision making.

Work	Symmetry locus	Primary use	Process assumption or information source	Operational relation
Liu & Popplestone (1990)	Assembly component geometry	Kinematic and spatial constraint inference	Component feature symmetries reduce constraint satisfaction problem (CSP) combinatorics	The assembly method uses component symmetry for constraint inference; our framework transports sequential process relations
Fox & Long (1999)	Planning problem objects and actions	Detect and exploit symmetric search structure	Exact problem symmetry exposes equivalent choices	The planner reduces equivalent choices; our framework transports process directionality obtained from independent evidence
Fox et al. (2005)	Abstract planning problem; objects and actions that are almost symmetric	Proactive action ordering in forward search	Guidance is induced from symmetry revealed by abstraction and prior plan actions	The planner uses abstract symmetry for action ordering; our framework transports independently identified process relations
Dynamic SAT symmetry handling (Devriendt et al., 2012; 2017)	Boolean theory and literal symmetries	State-dependent propagation and clause learning	Full problem symmetry preserves logical consequences; the current assignment determines when transported consequences restrict search	SAT transports logical consequences under formula symmetry; our framework transports process evidence and refines proposal ranks
SayCan (2023)	Embodied state and skill set	Ground language model skill selection in current affordances	Value functions score whether each pretrained skill is currently executable and are reevaluated through execution	SayCan reevaluates skill affordances after state changes; our framework adds terminal transport of independently identified process structure
SymPlan (2023)	Path planning Markov decision process (MDP) and value iteration operator	Equivariant differentiable planning	Relevant state and action transformations are symmetries of the planning problem	SymPlan structures planner computation by equivariance; our framework separates process evidence, terminal correspondence, and state refinement
Equivariant Motion Manifold Primitives (2023)	Manifold of tasks and solution trajectories	Equivariant trajectory family learning	Task symmetry supports equivariant motion families	The motion model learns equivariant trajectory families; our framework combines terminal correspondence, process evidence, and state residuals
Partial equivariant RL (2026)	State-action regions of an MDP	Selective invariant and standard Bellman backups	Symmetry validity varies across local process regions	Partial equivariant RL adapts Bellman backups locally; our framework uses fixed terminal correspondence and updated decision relevance
Cross-space symmetry composition (2026)	Configuration and task spaces	Lifting, descending, and composing symmetries for jointly equivariant policies	Multiple policy symmetries are composed across spaces	The policy composes configuration and task symmetries; our framework transports directed process evidence through terminal correspondence
Canonical diffusion (2026a)	Orbit representatives of invariant targets	Canonical representative selection	Invariant target distribution is modeled on a canonical slice	Canonical diffusion selects an orbit representative; our framework uses state residuals to update the transported structure's decision value
Carrier framework	Terminal outcomes plus external process evidence	Transport, refinement, and certification for verified construction	Directionality is independently grounded; exact terminal correspondence supplies transport	Our framework fixes transported structure within an episode, updates its decision relevance, and certifies actions with a fixed verifier

C Extended Empirical Evidence

C.1 SymBuild Baselines and Ablations in Three Construction Domains

Table 8 compares SymBuild, its single-channel ablations, and permuted transport. Alternative aggregation and planning methods appear in Appendices C.5 and C.6. The process and state controls are, respectively, PCS-ICP and outsideness for CAD, combined learned and observed process relations and transported observed relations for Mini-Programs, and the process prior and best fit for exact-fill packing. CAD correspondence evidence comes from the aligned versus permuted intervention on disjoint objects in Table 2(b); the corresponding cells in Table 8 are marked with an em dash.

Algorithm 1 SymBuild: statewise residual rank meet

Require: target X, correspondence g, evidence R_{ref}, state s_0, verifier V, call cap B, deterministic key κ_s

$R_T \leftarrow \textsc{Transport}(R_{\mathrm{ref}}, g)$; $s \leftarrow s_0$; $q \leftarrow 0$; $\tau \leftarrow ()$
while s is incomplete and $q < B$ **do**
  $A_s \leftarrow$ remaining actions; $r_T^s \leftarrow \textsc{RestrictRank}(R_T, A_s)$
  $r_X^s \leftarrow \textsc{ResidualRank}(s, A_s)$; $m_s(a) \leftarrow \min\{r_T^s(a), r_X^s(a)\}$
  $Q \leftarrow \textsc{Sort}(A_s, (m_s(a), \kappa_s(a)))$
  accepted $\leftarrow$ false
  while $Q \neq \emptyset$ and $q < B$ **do**
    $a \leftarrow \textsc{PopFirst}(Q)$; $y \leftarrow V_s(a)$; $q \leftarrow q + 1$
    if $y = 1$ **then**
      $s \leftarrow F(s, a)$; $\tau \leftarrow \tau \circ a$; accepted $\leftarrow$ true; **break**
    end if
  end while
  if $\neg$accepted and $q < B$ **then**
    return $(\tau, q,$ candidates exhausted$)$
  end if
end while
if s is complete **then**
  return $(\tau, q,$ completed$)$
else
  return $(\tau, q,$ budget exhausted$)$
end if

Table 7: **Formal SymBuild evaluation coverage in the three construction domains.** Counts refer to confirmation objects; each seed is evaluated for every applicable method and budget. n denotes parts, events, or remaining items.

Domain	Objects	Seeds	Setting	Budget grid	Verifier resource
CAD calls	72	2 sensor	ID, 5–20 parts	6 values, $1.25n$–$4n$	proposal calls
CAD work	48	2 sensor	ID, 5–20 parts	6 values, 1.25–4	dense sweep steps
Mini-Program	480	2 data	ID, length OOD	5 values, $1n$–$2n$	precedence queries
Exact-fill packing	480	2 data	ID, size OOD	6 values, $1n$–$3n$	exact extension queries

Table 8: **Complete SymBuild comparison and ablation matrix.** Each cell gives anytime AUC first and final verified success second, both in percent (↑). Static is the matched static control for statewise refresh. Process and State retain only the corresponding information channel; Permuted disrupts transport while preserving the remaining interface. Program OOD and Packing OOD denote the length and size OOD splits. Best and second-best distinct AUCs among the displayed methods are bold and underlined; ties share formatting.

Panel	N	SymBuild	Static	Process	State	Permuted
CAD calls	72	**81.60/94.62**	78.73/89.93	<u>79.57/89.58</u>	73.12/91.49	–
CAD work	48	**98.70/98.70**	<u>91.93/91.93</u>	87.50/87.50	85.63/85.94	–
Program ID	240	<u>85.08/100</u>	72.67/100	84.83/100	**85.17/100**	15.83/37.50
Program OOD	240	<u>64.67/97.50</u>	42.92/83.33	63.58/97.08	**65.83/97.50**	1.25/4.58
Packing ID	240	**57.92/100**	49.38/99.58	<u>50.21/100</u>	38.26/100	13.33/68.75
Packing OOD	240	**34.03/95.00**	25.35/83.33	<u>28.40/93.75</u>	21.67/99.17	3.68/20.42

Table 9 reports paired uncertainty for the CAD mechanism contrasts in Main Table 2(b); Appendix D specifies the panel construction and resampling rules.

Table 9: **Paired uncertainty for Main Table 2(b).** Success gain is the named condition minus its comparator; call saving is the comparator minus the named condition in calls per part, so positive values favor the named condition. Intervals are paired 95% CIs from 10,000 bootstrap draws over assemblies; Appendix D specifies the panel construction and resampling units. All reported confidence intervals for call savings are positive.

Test	Condition	Comparator	Success gain (pp ↑) [95% CI]	Call saving ↑ [95% CI]
Algorithm	SymBuild	PCS-ICP	$+5.00\,[-.42, 10.83]$	$+.143\,[.050, .240]$
Algorithm	SymBuild	outsideness	$+7.71\,[2.29, 13.54]$	$+.168\,[.066, .279]$
Timing	SymBuild	Static	$+4.38\,[-.42, 9.58]$	$+.152\,[.064, .246]$
Alignment	aligned	permuted	$+16.25\,[8.33, 25.00]$	$+.637\,[.501, .782]$

C.2 SymBuild Replication by Group

Table 10: **AUC replication for SymBuild relative to Static.** Values report the AUC difference, computed as SymBuild minus Static; brackets are paired 95% intervals. Within each row, paired differences and intervals follow the order of the two groups from left to right.

Domain and resource	Group	Gap (pp ↑)	95% CI
CAD calls	sensor 701 / 709	$+2.55/+3.18$	$[-.29, 5.44]/[.12, 6.31]$
CAD calls	5–8 / 9–20 parts	$+.61/+3.99$	$[-5.12, 6.34]/[1.00, 7.29]$
CAD work	sensor 811 / 823	$+6.77/+6.77$	$[1.04, 14.06]/[1.04, 14.06]$
CAD work	5–8 / 9–20 parts	$+3.91/+8.20$	$[0, 11.72]/[.39, 17.97]$
Program ID	seed 20270117 / 20270129	$+10.50/+14.33$	$[6.50, 14.50]/[10.17, 18.67]$
Program OOD	seed 20270117 / 20270129	$+24.50/+19.00$	$[19.50, 29.33]/[14.67, 23.33]$
Packing ID	seed 20271103 / 20271117	$+8.06/+9.03$	$[5.83, 10.42]/[6.81, 11.25]$
Packing OOD	seed 20271103 / 20271117	$+7.36/+10.00$	$[5.28, 9.58]/[7.64, 12.64]$

C.3 Correspondence Fidelity and Approximate Verification

This section expands Section 4.4 with the graded correspondence fidelity intervention and the complete approximate verifier sweep.

Mini-Program intervention on correspondence fidelity. To diagnose correspondence fidelity, we progressively mismatch action identities while reusing the same frozen 480-object confirmation population, checkpoint, observed expert, verifier, and budget grid. For a requested fraction $f \in \{0, .1, .25, .5, 1\}$, let M be the non-root event identities and set $m = \mathrm{round}(f|M|)$, except that a singleton move is promoted to two identities when possible. A sample of m identities fixed by the seed is cyclically shifted by one position and used to conjugate the learned precedence matrix; all unsampled identities remain fixed. Thus the realized moved fraction is measured directly as $m/|M|$. The perturbation seed is a deterministic function of data seed, split, object index, and requested mismatch level. Intervals for the difference between clean and mismatched AUC use 10,000 paired bootstrap draws on the shared objects.

At the requested 10% mismatch, moved fractions are 12.78% on ID and 10.21% on OOD because small graphs require a swap of two nodes; all other means are within $.74$ points of the request. The difference in AUC between clean and mismatched conditions is already positive at the first nonzero level: $9.75\,[7.75, 11.83]$ points on ID and $15.08\,[12.67, 17.50]$ on length OOD. The monotone frontier in Figure 5 shows that transport value changes systematically with correspondence fidelity.

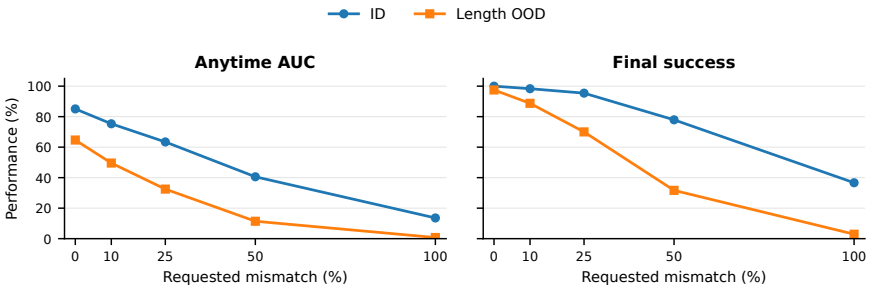

Figure 5: **Transport value decreases as correspondence fidelity degrades.** Exact AUC and final success values appear in Main Table 3(a); both decrease as transported action identities are increasingly mismatched.

C.3.1 Approximate Verification

Table 11: **Complete stochastic verifier robustness test.** True AUC uses hidden exact validity. FC denotes false certification. ΔAUC is SymBuild minus Static, and FC pairs follow the same order. Higher values are better for true AUC and ΔAUC (↑); lower values are better for FC (↓). Values are averaged over 20 noise seeds within each 240-object split before paired bootstrap.

p	SymBuild true AUC	Static true AUC	ΔAUC [95% CI]	FC (SymBuild / Static)	Seeds (objects)
False positive – ID					
.01	**83.43**	69.81	+13.62 [10.53, 16.76]	**2.60**/5.38	20 (240)
.025	**81.36**	66.08	+15.28 [11.92, 18.73]	**5.79**/12.21	20 (240)
.05	**78.27**	60.80	+17.47 [13.70, 21.25]	**10.67**/21.56	20 (240)
.10	**72.72**	52.09	+20.62 [16.03, 25.20]	**18.92**/36.46	20 (240)
False positive – Length OOD					
.01	**60.75**	38.81	+21.94 [18.68, 25.26]	**8.10**/12.42	20 (240)
.025	**54.63**	33.24	+21.39 [18.10, 24.63]	**19.88**/28.17	20 (240)
.05	**46.78**	26.30	+20.48 [17.18, 23.76]	**34.12**/48.12	20 (240)
.10	**35.62**	17.72	+17.90 [14.61, 21.23]	**52.85**/70.48	20 (240)
False negative – ID					
.01	**81.04**	69.64	+11.40 [8.59, 14.16]	0/0	20 (240)
.025	**74.96**	65.01	+9.95 [7.39, 12.57]	0/0	20 (240)
.05	**66.97**	58.33	+8.65 [6.25, 11.13]	0/0	20 (240)
.10	**52.88**	45.97	+6.92 [4.84, 9.06]	0/0	20 (240)
False negative – Length OOD					
.01	**61.28**	40.74	+20.54 [17.35, 23.73]	0/0	20 (240)
.025	**55.97**	38.09	+17.88 [14.94, 20.85]	0/0	20 (240)
.05	**48.57**	33.56	+15.01 [12.36, 17.75]	0/0	20 (240)
.10	**36.38**	24.98	+11.40 [9.16, 13.60]	0/0	20 (240)

On the same 480 confirmation objects, each combination of object, completed set, candidate action, regime, and noise seed receives a deterministic noise draw shared across paired methods. For $p \in \{.01, .025, .05, .10\}$, false positives accept invalid actions and mark them complete, while false negatives reject valid actions and leave the state unchanged, each with probability p; all other responses match the exact verifier. A hidden flag tracks true prefix validity. We average over noise seeds 0–19 for each object and condition before 10,000 paired bootstrap draws. Bootstrap sampling uses base seed 61000 with deterministic offsets for split, noise regime, noise level, and metric. Completion marks all events complete; true success additionally requires every accepted action to be valid; false certification is completion without true success.

C.4 Complete Anytime Frontiers

Table 12 gives the exact success percentages for rank meet with statewise refresh and its Static control underlying Figure 3. The ablations using a single channel appear in Main Table 2(a), while the independent CAD alignment intervention is summarized in Main Table 2(b).

Table 12: All budget points for rank meet with statewise refresh and its Static control. Budgets were predeclared and normalized by the number of parts, events, or remaining items, or by one dense sweep per part. Best values are bold and second-best values are underlined; ties are unmarked.

Domain and split	Predeclared budgets	Verified success ↑ at successive budgets
CAD calls	1.25, 1.5, 2, 2.5, 3, 4	SymBuild: **68.92**, **74.13**, 78.65, **83.16**, **90.10**, **94.62** Static: 64.93, 71.35, 78.65, 82.47, 85.07, 89.93
CAD work	1.25, 1.5, 2, 2.5, 3, 4	SymBuild: **98.70**, **98.70**, **98.70**, **98.70**, **98.70**, **98.70** Static: 91.93, 91.93, 91.93, 91.93, 91.93, 91.93
Program ID	1, 1.25, 1.5, 1.75, 2	SymBuild: **48.33**, **82.92**, **94.58**, **99.58**, 100.00 Static: 27.92, 55.42, 82.92, 97.08, 100.00
Program OOD	1, 1.25, 1.5, 1.75, 2	SymBuild: **8.33**, **47.92**, **78.75**, **90.83**, **97.50** Static: 5.42, 18.33, 40.83, 66.67, 83.33
Packing ID	1, 1.125, 1.25, 1.5, 2, 3	SymBuild: 9.58, **31.25**, **44.17**, **72.92**, **89.58**, **100.00** Static: 9.58, 22.50, 32.92, 50.42, 81.25, 99.58
Packing OOD	1, 1.125, 1.25, 1.5, 2, 3	SymBuild: 0.00, **4.58**, **16.25**, **30.42**, **57.92**, **95.00** Static: 0.00, 2.50, 5.42, 16.67, 44.17, 83.33

CAD work at finer budget resolution. Supplementary diagnostic budgets resolve the completion thresholds before the original CAD work grid begins. Table 13 gives verified success over the same 384 paired evaluation conditions at this finer resolution.

Table 13: **Finer-resolution CAD work diagnostic.** Work budgets are full sweep equivalents per part. Entries reuse the original 48 assemblies, two sensor seeds, four lifts, and $4N$ cap on proposal calls. Success percentages are measured over all evaluation conditions; differences use 10,000 paired bootstrap draws over assemblies after averaging sensor and lift conditions.

Work budget b (sweeps per part)	SymBuild success (%)	Static success (%)	Δ success (pp) [paired 95% CI]
1.000	37.24	**38.54**	$-1.30\,[-12.76, 10.16]$
1.025	**77.34**	68.23	$+9.11\,[-1.04, 20.05]$
1.050	**88.80**	76.04	$+12.76\,[3.13, 23.44]$
1.100	**97.14**	84.64	$+12.50\,[3.65, 22.40]$
1.150	**98.70**	90.63	$+8.07\,[2.34, 15.36]$
1.200	**98.70**	91.93	$+6.77\,[1.30, 14.06]$
1.250 (original start)	**98.70**	91.93	$+6.77\,[1.30, 14.06]$

At $b = 1.05$ and 1.10, SymBuild raises verified success by 12.76 and 12.50 points, respectively. The two methods reach their completion plateaus by $b = 1.20$, with SymBuild completing 26 additional paired conditions. These diagnostics expose both higher completion probabilities at smaller budgets and greater completion coverage under the fixed proposal cap. On the original grid, each method's success curve is constant, so its AUC equals its plateau success and the tight, uniform, and relaxed interruption distributions give the same value.

C.5 Alternative Aggregation Beyond SymBuild

Development control for rank aggregation. The CAD development comparison favors rank meet in both success and verifier cost. The panel contains 60 objects with four fixed C_4 lifts each and holds both rank channels and the verifier fixed. Rank meet achieves 86.25% success at cost 1.547, compared with 80.00% at 1.752 for Borda count and 79.17% at 1.820 for rank join, the pointwise maximum of the two ranks. This joint improvement supports the choice of rank meet for confirmation (Section 4.5).

Packing development control. A disjoint panel of 80 instances uses seeds 314159 and 271828, with 40 instances per seed and 4–6 bins. It uses capacity 100, 25% prior corruption, the exact extension verifier, and the confirmation budget grid. Rank meet, Borda count, and rank join are compared using 10,000 bootstrap draws paired by instance with base seed 25101; rank meet is then used for confirmation.

Weighted sum selection. A disjoint 160-object Mini-Program development set uses seeds 20270203 and 20270211, with 40 objects per seed and split and identities disjoint from confirmation and earlier experiments. It evaluates weighted rank sums $wr_T + (1-w)r_X$ for $w \in \{.25, .50, .75\}$. The mean of ID and OOD development AUC selects $w = .25$, which is frozen for the 480-object confirmation panel. Refreshed and static variants share checkpoint, candidates, verifier, budgets, tie-breaking, and the bootstrap protocol at the object level.

Table 14: **Alternative aggregation confirmation.** Panel (a) compares rank meet with the weighted sum selected on development data for Mini-Programs; AUC, ΔAUC, and final success are better when higher (↑), while cost is better when lower (↓). Panel (b) reports the SayCanPay-adapted product rule and a parameter-free multi-heuristic key (MH key) on all six construction panels. Each cell reports AUC followed by final success, both as percentages.

(a) Weighted aggregation selected on development data

Split	Aggregator	Refresh AUC	Static AUC	ΔAUC [95% CI]	Final (refresh / static)	Cost (refresh / static)	Selection
ID	Rank meet	**85.08**	72.67	+12.42 [9.50, 15.33]	100/100	**.5719**/.6456	primary
ID	Weighted sum $w = .25$	**85.17**	71.67	+13.50 [10.50, 16.42]	100/100	**.5718**/.6530	development
Length OOD	Rank meet	**64.67**	42.92	+21.75 [18.33, 24.92]	**97.50**/83.33	**.6706**/.8057	primary
Length OOD	Weighted sum $w = .25$	**65.83**	41.67	+24.17 [20.75, 27.50]	**97.50**/82.08	**.6636**/.8132	development

(b) Alternative aggregators in the three construction domains

Panel	Product refresh	Product static	MH key refresh	MH key static
CAD calls	82.38/95.49	79.08/90.10	81.63/94.44	78.76/89.76
CAD work	98.18/98.18	93.75/93.75	98.70/98.70	91.93/91.93
Program ID	85.08/100	72.67/100	85.08/100	72.83/100
Program OOD	65.50/97.50	42.92/83.75	64.33/97.50	42.83/82.50
Packing ID	72.29/100	65.83/100	50.21/100	50.21/100
Packing OOD	47.92/98.33	41.39/99.17	28.40/93.75	28.40/93.75

Table 15: **Post-transition pattern with alternative aggregators.** Every pair agrees on its first accepted action. Sequence disagreement comes from matched execution traces; gaps compare refresh with the identical static aggregation. Product denotes the SayCanPay-adapted product rule; the Refresh only and Static only columns count verified completions unique to the refreshed and static variants, respectively.

Aggregator	Panel	Pairs	First same	Sequence differs	Refresh only / Static only	AUC gap ↑ [95% CI]
Rank meet	CAD calls	576	100%	96.35%	32 / 5	+2.86 [.12, 5.76]
Rank meet	CAD work	384	100%	93.75%	26 / 0	+6.77 [1.30, 14.06]
Rank meet	Packing ID	240	100%	67.50%	1 / 0	+8.54 [7.01, 10.14]
Rank meet	Packing OOD	240	100%	93.75%	29 / 1	+8.68 [7.01, 10.42]
Product	CAD calls	576	100%	97.40%	36 / 5	+3.30 [1.01, 5.79]
Product	CAD work	384	100%	94.79%	17 / 0	+4.43 [.52, 9.90]
Product	Program ID	240	100%	98.75%	0 / 0	+12.42 [9.50, 15.33]
Product	Program OOD	240	100%	100%	36 / 3	+22.58 [19.33, 25.83]
Product	Packing ID	240	100%	97.08%	0 / 0	+6.46 [4.51, 8.47]
Product	Packing OOD	240	100%	100%	0 / 2	+6.53 [4.58, 8.47]

Post-transition pattern with alternative aggregators. The product rule provides a second aggregation test of the same refresh mechanism in the three construction domains. Table 15 reports the paired post-transition pattern for rank meet and product variants under matched static controls.

C.6 Alternative Planning Methods

Common interface. At the search interface, r_T^s and r_X^s support multiple proposal methods: multi-heuristic scheduling keeps separate queues, partial-order causal-link (POCL)-style scheduling (McAllester & Rosenblitt, 1991) constrains an agenda, LazySP-style selection performs prefix search without verifier calls, and rank meet compiles the prefixes into one order. Section 4.5 tests these framework instantiations.

These controls use the same 480 Mini-Programs, frozen relation model, candidates, verifier, budgets, and deterministic tie order as the primary panel. Each data group contains 120 ID objects (3–5 leaves) and 120 length OOD objects (6–8 leaves). The cap is $2n$ verifier calls and anytime AUC averages success at $q/n \in \{1, 1.25, 1.5, 1.75, 2\}$. Paired intervals use 10,000 bootstrap draws paired by object; search and canonicalization comparisons use base seed 28101, while comparisons between refreshed

and fixed schedulers use base seed 28201. Deterministic offsets index split, contrast, metric, and budget.

MHA*-style scheduling and selection. A disjoint 160-object development set tests service ratios between the process and state queues 1:1, 1:2, 2:1, 1:3, 3:1 and selects 1:2 by mean ID/OOD AUC. For a ratio p:s, each cycle emits the next p unseen candidates ordered by process rank and then the next s candidates ordered by state rank, skipping duplicates; the selected ratio is frozen before confirmation.

The transported process rank and state residual rank define the two queues. The refreshed variant rebuilds both after every accepted event; the fixed counterpart filters the initial queues as events are completed without recomputing their ranks.

Canonical slice control. The exact terminal correspondence defines an orbit under identity and swap. For each program, the observable serialization of both representatives is formed from event types, numeric attributes, and observed edges; the lexicographically smaller representative is selected. Its canonical event positions provide the final tie resolution for the Static order produced by rank meet. Thus the control uses exact terminal canonicalization to reduce ambiguity in orbit labels while retaining the same observable information and verifier access as the primary method. State evidence is held at its initial value, isolating representative selection from refresh after transitions.

POCL-style partial-order agenda. An observable acyclic partial order is constructed from transported observed edges and learned edges of probability at least .5; any edge that would create a cycle is dropped. At each state, candidates with fewer remaining predecessors are served first; state and process ranks, followed by the shared deterministic tie order, resolve agenda ties. The refreshed variant recomputes those ranks after every accepted event. Its fixed counterpart updates which predecessors remain but retains the initial state and process ranks for tie resolution.

Weighted A* with LazySP-style replanning. A bounded best-first search proposes short candidate paths from the observable projected partial order and state rank using prefix expansion without verifier calls. Search width is 32, horizon is 4, and the weighted best-first coefficient is 2. The selected path contributes its first action to the shared verifier. Rejection invalidates that state-action choice and triggers replanning; acceptance advances the state and restarts planning from the successor.

Table 16: **Full comparison of search and canonicalization under matched interfaces.** AUC and final success are percentages. Selector time denotes mean Python proposal time per object; expansions count prefix expansions performed without verifier calls. All rows share the same 480-object population, observable information, verifier, and $2n$ cap.

	ID				Length OOD			
Method	AUC ↑	Success@$2n$ ↑	Selector ms ↓	Expansions ↓	AUC ↑	Success@$2n$ ↑	Selector ms ↓	Expansions ↓
SymBuild	85.08	100.00	5.5	0	64.67	97.50	11.2	0
Static	72.67	100.00	2.0	0	42.92	83.33	3.2	0
POCL-style refresh	85.67	100.00	9.1	0	66.33	97.92	21.8	0
POCL-style fixed	84.58	100.00	6.1	0	62.50	96.67	13.5	0
Canonical slice with static meet	72.50	100.00	8.3	0	42.92	82.92	19.9	0
MHA*-style 1:2 refresh	85.92	100.00	8.4	0	65.83	97.08	20.4	0
MHA*-style 1:2 fixed	86.33	100.00	5.1	0	60.83	95.00	11.9	0
Weighted A* with LazySP-style replanning	80.67	99.58	94.2	391.7	60.42	96.25	193.3	609.6

On length OOD, refresh reduces capped normalized verifier cost by 2.35 points for POCL-style (95% CI $[1.81, 2.91]$) and 3.26 for MHA*-style ($[2.11, 4.42]$). The MHA*-style costs are .5652 for refresh and .5641 for the fixed variant on ID, and .6635 and .6962, respectively, on length OOD.

C.7 GRN and Population-Based Search

Target episodes. The official Panda OOD scenes (Bouhsain et al., 2025) are converted into target removal episodes with one deterministic annotation verifier shared by all methods. From the empty removed set, breadth-first search enumerates states reachable through accepted removals; a movable object is a target if it is removed on at least one reachable transition. This yields 129 targets from 20 OOD10 scenes, 432 from 50 OOD15 scenes, and 574 from 50 OOD20 scenes, for 1,135 targets total.

Candidate verification and transitions. Every remaining movable object contributes five candidates in the fixed grasp order Top, Front, Rear, Right, Left. Candidate (o, g) is accepted iff the object remains annotated, no remaining object names o as its parent in the support frame, the official inverse kinematics (IK) annotation is feasible, and the sum of geometric obstruction failure ratios from remaining blockers is below $1 - 10^{-12}$. Acceptance removes the object and clears rejected candidates; rejection leaves the scene unchanged and suppresses only that object and grasp pair until the next accepted removal.

Score construction. The fixed GRN model assigns a grasp feasibility score to each candidate object and grasp pair. Process utility comes from the current directed obstruction and support graph: reverse transitive depth from the target gives larger values to remaining blocker ancestors and -1 to objects outside the target ancestry, with each object value copied to its five grasps. Each channel is independently normalized to z scores over the current $(\text{object}, \text{grasp})$ pairs; a channel with standard deviation below 10^{-12} is mapped to zero. Ranking by GRN or process uses the corresponding normalized channel, and combined ranking uses their unweighted sum.

Static and refresh updates. GRN static caches the initial raw GRN values. Combined static caches both initial raw channels. After accepted removals, removed objects are filtered out and the cached channels are normalized again over the remaining candidates. GRN refresh recomputes the raw GRN values after each accepted removal. Combined refresh recomputes both raw channels before normalization. Candidates are ordered by descending score with a shared deterministic tie order.

Budgets and statistical evaluation. All methods use a $5N$ verifier call limit. Table 5 records issued calls on success and $5N + 1$ on failure. Direct ranking uses tie seeds 0–4 at all three scales. Results are averaged over seeds for each target before inference, which uses 20,000 bootstrap draws clustered by scene and sign permutations with seed 20260809.

Population-based search. Our adaptation follows the population-based search strategy of Mishra et al. (2026) under the same candidate, score, transition, and verifier interface. Each of two rounds samples 12 imagined sequences of horizon at most four from a softmax with temperature .8; imagined removals use no verifier calls. A sequence receives its log sampling probability divided by sequence length plus a bonus of 2 if it reaches the target, and the top 25% form the elite set. Between rounds, counts of first actions in the elite set are weighted by inverse rank to form a guide distribution; its log probability enters the score at the first step with weight .5. After the second round, the first action with the largest exponentiated elite vote is passed to the shared verifier. CDGS–GRN uses the refreshed GRN channel alone. CDGS–static uses the static combined channels above. CDGS–refresh uses the refreshed combined channels. Population-based search uses seeds 0–4 at all three scales. The seed sets are shared across variants; samples follow each variant's score distribution. Target averaging and inference match the direct methods, including clustering by scene.

Ordinal fusion with learned state scores. The ordinal methods use the same 1,135 reachable targets, three OOD scales, candidate sets, fixed verifier, and $5N$ cap as the learned scoring methods above. Here N counts initially movable objects. The process structure is the scene's annotated obstruction and support graph, aligned by the identity correspondence on the shared object indices. For each full remaining candidate set A_s, negative process and GRN utilities are separately converted to one-based average weak ranks r_T^s and r_X^s, with exact equality defining ties. Rank meet orders candidates lexicographically by

$$\big(\min(r_T^s, r_X^s),\ r_T^s + r_X^s,\ \max(r_T^s, r_X^s),\ \pi(a)\big).$$

Borda and rank product use the respective keys

$$(r_T^s + r_X^s, \max(r_T^s, r_X^s), \pi(a)),$$
$$(r_T^s r_X^s, r_T^s + r_X^s, \max(r_T^s, r_X^s), \pi(a)).$$

These parameter-free rules provide three ordinal realizations using identical information. Static retains both initial raw channels, filters removed objects, and reconstructs ranks on the current full candidate set. Refresh recomputes raw utilities after each accepted removal before constructing ranks. Rejection masks the queried object–grasp pair while retaining the current state's scores and ranks. The process-only controls follow the corresponding raw-score caching and refresh rules.

Matched evaluation and inference. The six selectors are evaluated in paired Static and refresh variants with seeds 0–4, giving 645, 2,160, and 2,870 conditions per selector and variant on OOD10, OOD15, and OOD20, respectively, and 68,100 runs in total. Candidate priorities are sampled once for each scene, target, and seed and shared across methods; object and grasp indices resolve remaining ties. The official checkpoint is fixed, and the ordinal rules require no additional training or parameter selection. Bootstrap and sign permutation inference use the mean paired difference across targets after averaging the five seeds within each target. With random seed 20260809, bootstrap inference forms 95% percentile intervals from 20,000 resamples of entire scenes and averages all targets in each resample. The two-sided sign permutation test uses 20,000 assignments, applying one shared sign to all target differences within each scene. Table 17 reports query scores, completion rates, and paired inference for all six selectors. The tail query score reports the 95th percentile over conditions pooled across targets and seeds, using the same failure penalty as the mean score.

Table 17: **Complete direct scoring and ordinal comparisons in GRN target removal.** Mean query scores and paired refresh minus static gaps use the common $5N$ cap. Completion percentages list static first and refresh second. The best and second-best query scores among all twelve conditions at each scale are bold and underlined; ties share emphasis. The reported p values come from a two-sided sign permutation test applied at the scene level.

Split	Selector	Static ↓	Refresh ↓	Δ [95% CI]	Completion (%)	p
OOD10	GRN	5.457	5.132	-0.326 $[-0.743, 0.066]$	100.00, 100.00	0.17569
OOD10	Process	9.454	8.326	-1.129 $[-1.898, -0.464]$	97.98, 98.91	0.00230
OOD10	Combined	3.349	**2.597**	-0.752 $[-1.279, -0.270]$	100.00, 100.00	0.00645
OOD10	Rank meet	5.352	4.747	-0.605 $[-1.433, -0.025]$	99.22, 100.00	0.09685
OOD10	Borda	3.073	<u>2.654</u>	-0.419 $[-0.790, -0.118]$	99.22, 100.00	0.02385
OOD10	Rank product	3.794	3.205	-0.589 $[-1.370, 0.025]$	100.00, 100.00	0.15639
OOD15	GRN	8.213	6.734	-1.479 $[-2.053, -0.978]$	100.00, 100.00	0.00005
OOD15	Process	15.492	12.301	-3.191 $[-4.202, -2.256]$	97.18, 98.84	0.00005
OOD15	Combined	4.523	**3.349**	-1.174 $[-1.500, -0.865]$	100.00, 100.00	0.00005
OOD15	Rank meet	7.097	6.127	-0.970 $[-1.472, -0.560]$	100.00, 100.00	0.00005
OOD15	Borda	4.825	<u>3.576</u>	-1.249 $[-1.650, -0.888]$	100.00, 100.00	0.00005
OOD15	Rank product	5.575	4.356	-1.219 $[-1.614, -0.869]$	100.00, 100.00	0.00005
OOD20	GRN	11.027	8.028	-2.999 $[-4.372, -1.879]$	99.83, 100.00	0.00005
OOD20	Process	21.912	15.755	-6.157 $[-8.229, -4.231]$	96.45, 98.85	0.00005
OOD20	Combined	6.332	**3.873**	-2.459 $[-3.225, -1.774]$	100.00, 100.00	0.00005
OOD20	Rank meet	9.319	7.608	-1.711 $[-2.608, -0.995]$	100.00, 100.00	0.00005
OOD20	Borda	7.251	**3.873**	-3.378 $[-4.873, -2.115]$	99.13, 100.00	0.00005
OOD20	Rank product	7.364	<u>4.898</u>	-2.466 $[-3.376, -1.704]$	100.00, 100.00	0.00005

Verification tails and post-transition behavior. Matching the first action locates the benefit of refresh in subsequent decisions (Table 18). Ordinal rank meet reduces mean query scores by 11.30%, 13.67%, and 18.36% on OOD10, OOD15, and OOD20, respectively. Its tail query score falls from 16 to 10 on OOD10, from 19 to 15 on OOD15, and from 25 to 17 on OOD20, showing that the benefit extends to difficult episodes. Refresh also completes five OOD10 conditions left unfinished by Static.

The transported process channel complements learned feasibility: refreshed rank meet improves on GRN refresh by 0.606 queries $[0.308, 0.956]$ on OOD15 and 0.420 queries $[0.191, 0.698]$ on OOD20. Combined and Borda combine target and grasp evidence additively; rank meet keeps candidates favored by either source early in the order. Refresh gains across all six selectors support statewise refinement under these different aggregation choices, with Combined achieving the lowest mean query scores at all three scales.

Table 18: **Execution diagnostics for the matched GRN selectors.** Scored states and logical GRN forwards list static first and refresh second, including initial scoring. Sequence disagreement and unique completion counts use all conditions across the five seeds. Refresh only / Static only counts completions unique to each variant. Every pair shares its first queried action. Tail query score is the 95th-percentile query score and lists Static first and refresh second.

Split	Selector	Scored states (static, refresh)	Logical GRN forwards (static, refresh)	Sequence differs (%)	Refresh only / Static only	Tail score
OOD10	GRN	4.00, 4.00	1.00, 4.00	42.64	0 / 0	16/14
OOD10	Process	1.99, 1.86	0.00, 0.00	27.29	7 / 1	36/29
OOD10	Combined	2.30, 2.00	1.00, 2.00	27.13	0 / 0	10/7
OOD10	Rank meet	3.95, 3.95	1.00, 3.95	39.53	5 / 0	16/10
OOD10	Borda	2.00, 1.91	1.00, 1.91	27.13	5 / 0	10/6
OOD10	Rank product	2.78, 2.59	1.00, 2.59	30.23	0 / 0	14/11
OOD15	GRN	5.10, 5.10	1.00, 5.10	69.44	0 / 0	30/19
OOD15	Process	2.74, 2.47	0.00, 0.00	36.76	37 / 1	61/44
OOD15	Combined	3.13, 2.64	1.00, 2.64	48.84	0 / 0	16/10
OOD15	Rank meet	5.04, 5.08	1.00, 5.08	68.98	0 / 0	19/15
OOD15	Borda	2.91, 2.58	1.00, 2.58	47.92	0 / 0	14/8
OOD15	Rank product	3.85, 3.44	1.00, 3.44	56.94	0 / 0	18/15
OOD20	GRN	6.67, 6.67	1.00, 6.67	76.31	5 / 0	39/18
OOD20	Process	3.27, 2.77	0.00, 0.00	41.78	69 / 0	88/58
OOD20	Combined	4.06, 3.14	1.00, 3.14	47.39	0 / 0	26/12
OOD20	Rank meet	6.59, 6.64	1.00, 6.64	76.31	0 / 0	25/17
OOD20	Borda	3.65, 2.96	1.00, 2.96	47.91	25 / 0	34/9
OOD20	Rank product	4.87, 4.17	1.00, 4.17	55.64	0 / 0	27/15

Ordinal guidance within population-based search. Crossing aggregation with statewise refinement tests whether ordinal process and feasibility evidence remain useful within sequence search. Combined and rank meet each have a Static and a refresh variant, using the same reachable targets, checkpoint, candidate interface, verifier, $5N$ cap, and population parameters specified above. For the full remaining candidate set, the ordinal sampling utility is

$$u^s_{\text{meet}}(a) = \text{zscore}_{A_s}(-\min\{r^s_T(a), r^s_X(a)\}),$$

where the two channels use one-based average weak ranks. Nonconstant utilities are standardized to unit variance, and tied minimum ranks receive equal utility. CDGS uses this utility for sequence sampling with the fixed temperature and applies its existing elite guide. Direct rank meet uses the deterministic ordering defined above.

Static retains both initial raw channels and reconstructs normalization or ranks on each remaining candidate set. Refresh recomputes both channels in realized and imagined removal states. Rejection masks the queried pair while preserving the realized state, and imagined removals consume no verifier calls. Sequence guidance can therefore change the first queried action through scores in imagined successor states.

For each scene, target, and seed, all four variants initialize the same pseudorandom stream. The population comparisons in Table 5 use method-specific streams. The crossed comparison retains seeds 0–4 at each scale and averages them within each target before 20,000 paired scene-clustered bootstrap resamples and sign permutations with seed 20260809. Query scores retain the $5N + 1$ failure penalty. Each policy's scoring demand is counted independently of prediction reuse across methods, targets, and seeds.

Refinement across sampling utilities. Table 19 shows refresh gains for both aggregation rules at every scale. Ordinal guidance reduces mean query scores by 16.66%, 12.48%, and 19.10% on OOD10, OOD15, and OOD20, respectively, and refreshed policies complete every condition. All six paired confidence intervals favor refresh. These results extend the refinement pattern from direct ordinal ranking to imagined sequence selection. Combined achieves the lowest query scores within this crossed comparison, illustrating the value of choosing an aggregation suited to the available signals.

Table 19: **Aggregation and refinement within CDGS-style population search.** The four policies share instances, five search seeds, initialization of the random stream, checkpoint, search parameters, verifier, and $5N$ call limit, giving 22,700 runs. Mean query scores include the $5N + 1$ failure penalty. Δ is refresh minus Static with paired scene-clustered 95% confidence intervals. The reported p values use a two-sided sign permutation test with one shared sign per scene. Completion lists Static first and refresh second. Bold and underlined query scores identify the best and second best among the four policies at each scale.

Split	Aggregation	Static ↓	Refresh ↓	Δ [95% CI]	Completion (%)	p
OOD10	Combined	4.116	**2.958**	-1.158 $[-2.105, -0.400]$	99.22/100.00	0.00070
OOD10	Rank meet	4.868	<u>4.057</u>	-0.811 $[-1.449, -0.239]$	100.00/100.00	0.01710
OOD15	Combined	5.769	**3.818**	-1.951 $[-2.538, -1.406]$	99.95/100.00	0.00005
OOD15	Rank meet	5.813	<u>5.088</u>	-0.725 $[-1.186, -0.301]$	100.00/100.00	0.00110
OOD20	Combined	7.505	**4.192**	-3.312 $[-4.168, -2.486]$	99.90/100.00	0.00005
OOD20	Rank meet	7.490	<u>6.059</u>	-1.431 $[-1.982, -0.964]$	99.90/100.00	0.00005

Table 20 separates neural scoring from the candidate positions evaluated during sequence sampling. Refresh uses state-dependent neural predictions to reduce verifier queries and sampled positions under both utilities. Ordinal guidance explores more scored states than Combined, making the aggregation choice consequential for the balance between neural computation and verifier effort.

Table 20: **Scoring effort under the crossed aggregation comparison.** Means use all combinations of targets and seeds. Logical GRN forwards count distinct neural inputs when predictions are cached by removal state within each episode. The count includes initial scoring. Static uses one initial forward. Scored positions count candidate positions in imagined sequences. Both metrics list Static first and refresh second, and each policy's requirements are counted independently of prediction reuse across evaluations.

Split	Aggregation	Logical GRN forwards (Static, refresh)	Scored positions (Static, refresh)
OOD10	Combined	1.00, 12.87	227.56, 140.09
OOD10	Rank meet	1.00, 43.67	305.73, 249.14
OOD15	Combined	1.00, 23.19	364.41, 213.73
OOD15	Rank meet	1.00, 88.84	426.13, 368.50
OOD20	Combined	1.00, 38.34	531.28, 260.45
OOD20	Rank meet	1.00, 141.02	574.77, 452.35

Computational cost of population-based search. Distinct scored removal states, including the initial state, measure the demand on the neural scorer; action positions evaluated in imagined sequences measure the additional search effort. During effectiveness evaluation, predictions for identical scenes and removed sets are reused across methods, targets, and seeds. Each method's scoring requirements are counted independently of this reuse. CDGS–GRN evaluates 43.73, 71.26, and 109.06 distinct GRN states per target on OOD10, OOD15, and OOD20, respectively. Combined refresh and CDGS–refresh both achieve 100% completion under the shared $5N$ cap. Their state counts in Table 5(c) show that direct ranking attains this coverage with substantially less neural scoring than population-based search.

Table 21: **Related methods and their roles in this paper.** GRN and SayCanPay-adapted numerical comparisons use matched candidates, verifier, and budgets. Planning Neural Operator (PNO) supplies a motion-planning reference.

Method	Venue	License	Native interface	Use in this paper
SayCanPay (Hazra et al., 2024)	AAAI 2024	not stated in the official repository	language actions with Can and Pay scores	matched refresh and static comparison using the product rule
GRN (Bouhsain et al., 2025)	ICLR 2025	MIT	3D task and motion planning graph; grasp, inverse kinematics, and geometric obstruction labels	official OOD scorer and comparison under matched search
PNO (Matada et al., 2025)	ICLR 2025	CC BY-NC-SA 4.0	continuous value field over maps and configurations	contextual comparison at the task and interface levels

C.7.1 Wall-Clock Verification Value

Matched timing protocol. Wall-clock evaluation measures all six selectors and both variants with the same candidates, scoring operations, annotation verifier, and $5N$ cap as Appendix C.7. We use the official GRN checkpoint and Panda OOD data (Bouhsain et al., 2025). At each scale, the first ten scenes in identifier order are considered, scenes with incomplete movable-object annotations are excluded, and reachable targets are ordered by object identifier. Retaining at most 100 episodes yields 61 on OOD10, 82 on OOD15, and 100 on OOD20. All methods share tie seed 0.

Measured execution time includes initial and subsequent scoring, candidate masking, sorting, annotation verification, and state transitions. Static retains initial raw scores and normalizes or ranks them on remaining candidates; refresh reuses scores after rejections and recomputes them after accepted transitions. Each method, episode, and repeat starts with an independent score cache. We average three CUDA-synchronized repeats after model warm-up and rotate method order across repeats. Model and data loading, target enumeration, warm-up, and auxiliary operations are excluded from timing.

From query savings to execution time. Let t_s, t_r be measured execution times and $q_s > q_r$ the mean issued verifier calls for Static and refresh. An additional verifier latency λ gives projected time $t + \lambda q$, so the break-even latency is $\lambda^\star = (t_r - t_s)/(q_s - q_r)$. This projection uses actual issued calls for every episode, whereas query effectiveness uses the failure-penalized score. Tables 23 and 24 give the decomposition and projections from these matched measurements.

Scoring overhead and verification cost. The measurements identify when verifier savings translate into faster execution. Combined reaches break-even at 12.55, 31.26, and 31.44 ms per query on OOD10, OOD15, and OOD20, respectively. Rank meet saves queries while scoring more removal states, giving corresponding thresholds of 35.51, 86.01, and 224.50 ms per query. Above each threshold, the saved verification time exceeds the added scoring cost (Table 24). Rank meet refresh and both Combined variants complete every timing episode; rank meet Static completes 98.36% on OOD10 and 100% on both larger scales.

Timing variability is summarized in Table 22. The standard deviations of mean compute time across repeats are small relative to the scoring overheads in Table 23.

Table 22: **Variability of wall-clock measurements.** Entries are standard deviations in milliseconds across the three repeats of mean compute time over timing episodes. Populations, scoring operations, and measurement boundaries match Table 23.

	Rank meet		Combined	
Scale	Static	Refresh	Static	Refresh
OOD10	0.249	1.167	1.299	1.025
OOD15	0.723	0.880	0.626	0.488
OOD20	1.335	2.821	0.655	1.657

Table 23: **Wall-clock decomposition for all matched selectors.** Compute includes initial scoring and is reported in mean milliseconds per episode. Queries are actual issued verifier calls. Paired entries list static first and refresh second. The best and second-best compute means at each scale are bold and underlined. Break-even gives the additional per-query verification latency at which refresh and static have equal projected time.

Scale	Selector	Compute (static, refresh) (ms/episode) ↓	Issued queries (static, refresh) ↓	Saved queries ↑	Break-even (ms/query)
OOD10	GRN	16.578, 54.497	5.131, 4.820	0.311	121.740
OOD10	Process	**0.214**, <u>0.237</u>	7.967, 7.213	0.754	0.031
OOD10	Combined	16.574, 27.681	2.951, 2.066	0.885	12.547
OOD10	Rank meet	17.637, 56.635	5.508, 4.410	1.098	35.506
OOD10	Borda	17.052, 27.893	2.902, 2.492	0.410	26.450
OOD10	Rank product	17.840, 37.514	3.869, 2.951	0.918	21.431
OOD15	GRN	22.904, 99.972	7.866, 6.537	1.329	57.978
OOD15	Process	**0.382**, <u>0.431</u>	16.195, 12.183	4.012	0.012
OOD15	Combined	24.388, 59.335	4.171, 3.053	1.118	31.262
OOD15	Rank meet	24.763, 103.432	6.598, 5.683	0.915	86.011
OOD15	Borda	25.246, 61.420	4.549, 3.744	0.805	44.943
OOD15	Rank product	24.592, 75.495	5.366, 4.280	1.085	46.899
OOD20	GRN	31.540, 172.465	9.260, 7.610	1.650	85.409
OOD20	Process	**0.473**, <u>0.514</u>	15.060, 10.970	4.090	0.010
OOD20	Combined	32.994, 80.463	4.810, 3.300	1.510	31.436
OOD20	Rank meet	34.273, 173.465	7.870, 7.250	0.620	224.503
OOD20	Borda	32.457, 72.484	4.580, 3.130	1.450	27.605
OOD20	Rank product	33.346, 102.912	5.740, 4.480	1.260	55.211

Table 24: **Projected end-to-end latency difference: refresh minus static.** Entries are milliseconds per episode; negative values favor refresh. Projection adds the stated additional latency to every actual verifier call. At each scale and latency, the best and second-best differences among selectors are bold and underlined.

Scale	Selector	1 ms/query	10 ms/query	50 ms/query	100 ms/query	500 ms/query	1000 ms/query
OOD10	GRN	37.61	34.80	22.35	6.77	-117.82	-273.56
OOD10	Process	**-0.73**	**-7.52**	**-37.68**	<u>-75.39</u>	-377.03	-754.08
OOD10	Combined	<u>10.22</u>	<u>2.25</u>	<u>-33.16</u>	**-77.42**	-431.52	-874.14
OOD10	Rank meet	37.90	28.01	-15.92	-70.84	**-510.18**	**-1059.36**
OOD10	Borda	10.43	6.74	-9.65	-30.14	-194.08	-399.00
OOD10	Rank product	18.76	10.49	-26.23	-72.13	<u>-439.34</u>	<u>-898.36</u>
OOD15	GRN	75.74	63.78	10.60	-55.86	<u>-587.57</u>	<u>-1252.20</u>
OOD15	Process	**-3.96**	**-40.07**	**-200.56**	**-401.17**	**-2006.05**	**-4012.15**
OOD15	Combined	<u>33.83</u>	<u>23.77</u>	<u>-20.95</u>	<u>-76.84</u>	-524.00	-1082.94
OOD15	Rank meet	77.75	69.52	32.94	-12.79	-378.65	-835.97
OOD15	Borda	35.37	28.13	-4.07	-44.31	-366.27	-768.70
OOD15	Rank product	49.82	40.05	-3.37	-57.63	-491.78	-1034.46
OOD20	GRN	139.28	124.43	58.43	-24.07	-684.07	<u>-1509.07</u>
OOD20	Process	**-4.05**	**-40.86**	**-204.46**	**-408.96**	**-2044.96**	**-4089.96**
OOD20	Combined	45.96	32.37	-28.03	-103.53	<u>-707.53</u>	-1462.53
OOD20	Rank meet	138.57	132.99	108.19	77.19	-170.81	-480.81
OOD20	Borda	<u>38.58</u>	<u>25.53</u>	<u>-32.47</u>	<u>-104.97</u>	-684.97	-1409.97
OOD20	Rank product	68.31	56.97	6.57	-56.43	-560.43	-1190.43

D Experimental Protocols and Environment

Data provenance and domain construction. CAD objects come from the official Automated Sequence Planning (ASAP) training partition (Tian et al., 2024); each confirmation panel is disjoint

from earlier CAD experiments in both object identity and mesh content before deterministic selection. Mini-Programs are generated deterministically from the data seed and latent program specification, with confirmation splits that are disjoint in identity. Exact-fill packing instances are generated deterministically from the data seed, split, weights, true groups, the transport mapping from groups to bins, and visible anchors; confirmation IDs are pairwise disjoint and separate from development. Generation and ranking rules for each domain follow below.

D.1 Shared Evaluation and Statistics

Paired comparisons. The six primary CAD, Mini-Program, and packing panels use $10{,}000$ bootstrap draws paired at the object or assembly level; GRN and CDGS-style intervals use $20{,}000$ resamples clustered by scene after averaging search seeds for each target within its scene. Reported differences and projections are computed from unrounded values. The construction-domain interruption grids are summarized in Table 7; Appendix C.7 specifies the GRN cap.

Rank scales and budget evaluation. The theory uses ranks beginning at one. Refresh in CAD and packing applies the same positive affine normalization to both current ordinal ranks, preserving meet order and the rank threshold identity. Algorithm 1 counts proposal calls. Work-budget success requires a valid completion within the cumulative sweep threshold, evaluated from trajectories obtained under the original call cap.

D.2 CAD Assembly

The CAD evaluation panels in Table 25 use the dense 96-step MuJoCo (Todorov et al., 2012) verifier based on a straight convex-hull sweep, with escape factor 1.5. Table 25 collects their populations, sensor seeds, proposal caps, and bootstrap seeds. Within an assembly, sensor and lift conditions are averaged before paired resampling. The Algorithm and Timing contrasts share their 120 assemblies. The other evaluation panels use disjoint assemblies as specified below.

Table 25: **CAD evaluation panels.** Each evaluation assembly is tested under all four C_4 lifts. Population entries specify the parts quotas where defined. Each panel uses 10,000 bootstrap draws paired by assembly after averaging its sensor and lift conditions. Alignment training and validation populations are described in the learned residual protocol.

Panel	Evaluation population	Sensor seeds	Proposal cap	Bootstrap seed
Calls	72 assemblies: 24 with 5–8 parts and 48 with 9–20 parts	701, 709	$\lceil 4N \rceil$	22101
Algorithm	120 assemblies: 24 with 5–8 parts, 48 with 9–20 parts, and 48 with 21–50 parts	389	$\max(3N, 24)$	14801
Timing	Same 120 assemblies as Algorithm	503	$\max(3N, 24)$	14901
Alignment	60 evaluation assemblies	613	$\max(3N, 24)$	21501
Work	48 assemblies: 16 with 5–8 parts and 32 with 9–20 parts	811, 823	$4N$	23101

Correspondence and geometric controls. The external CAD correspondence controls use the official ASAP assembly assets. Assemblies are parsed either from archive meshes already expressed in assembled terminal coordinates or from public repository examples with explicit final poses. The development correspondence set includes finite, parseable assemblies with 5–50 parts. A deterministic partition based on assembly identity assigns 80% to training and 20% to validation, and augmented views inherit the assembly split. Terminal equivalence is the tabletop C_4 orbit formed by yaw rotations of 0°, 90°, 180°, and 270° about the fixed world vertical axis. Each correspondence view independently permutes part identities, applies a random $SE(3)$ transform in the sensor frame and one of these C_4 lifts, and renders directional partial observations with 55% retained visibility, at most 256 points per part, and Gaussian coordinate noise 0.002. The gravity and tool directions expressed in the sensor frame provide the process coordinate system used by the geometric controls.

PCS-ICP denotes iterative closest point (ICP) correspondence in this process coordinate system with Hungarian assignment. It first expresses the partial observations in the frame defined by the

tool and gravity directions, evaluates all four C_4 representatives, and runs up to four translation-only trimmed ICP refinements per representative, retaining the closest 80% of point correspondences at each refinement. Trimmed one-way Chamfer costs from partial to full geometry define the matrix between observed and canonical parts; Hungarian assignment gives a one-to-one part correspondence, and the representative with minimum mean assignment cost is selected. The recovered lift and assignment transport the canonical CAD geometry into the target frame. At each state defined by the remaining parts and for each fixed world direction, a directional blocker relation is recomputed on the predicted parts. The transported process score for a candidate is its number of predicted blockers plus a -0.01 tie-break based on normalized outward projection, with lower scores queried first. The **outsideness** control adapts ASAP's outside-in part heuristic (Tian et al., 2024) to the shared action set that pairs each part with a removal direction: using the same centers aligned with the process frame, candidate actions, and verifier, it scores part i along removal direction d by $-c_i^\top d$. Both controls share verifier acceptance and action semantics, isolating the proposal information.

Main CAD anytime confirmation on 72 objects. The primary CAD panel uses 72 ASAP training assemblies that are disjoint in both identity and content, with 5–20 parseable parts and a recoverable assembled configuration. A deterministic ordering based on assembly identity selects the parts quotas in Table 25. Its two sensor seeds define base partial views; each assembly paired with each sensor seed is evaluated under all four forced C_4 lifts, yielding $72 \times 2 \times 4 = 576$ conditions per method. Views follow the partial-observation settings in the correspondence protocol above. One shared calibration rotation perturbs the process axes defined by the tool and gravity directions by a Gaussian angle with mean zero and standard deviation 5°. Actions pair each remaining part with one of the six world directions $\pm x, \pm y, \pm z$.

For a set R of remaining parts, PCS-ICP supplies the recovered lift and the assignment from observed to canonical parts used to transport the reference CAD. For action (i, d), the transported process score is the number of predicted directional blockers plus the outward tie term

$$s_T(i, d; R) = |\mathrm{blockers}(i, d; R)| - 0.01 \, \frac{\langle c_i, d\rangle}{\max\{\mathrm{range}_{j\in R}\langle c_j, d\rangle, 10^{-8}\}},$$

where c_i is the transported predicted center. The local geometric score is $s_X(i, d; R) = -\langle x_i, d\rangle$, where x_i is obtained by centering the observed part centers, expressing them in the process frame aligned with the tool and gravity, and dividing all centers by their maximum radius (with a 10^{-8} floor). With $m = 6|R|$ current actions, each score channel is independently converted to average weak ranks and normalized to $[0, 1]$ by subtracting one and dividing by $\max(m - 1, 1)$. SymBuild's rank meet orders actions by $\min(r_T, r_X) + 10^{-6} r_T + 10^{-9} r_X$, followed by part and direction indices as the final deterministic tie-break. After an accepted removal, the blocker relation and both current ranks are recomputed on the reduced action set; a rejected proposal leaves the state unchanged and is skipped for that state. The Static control computes the complete fused ordering once at the initial state and subsequently only removes actions whose parts have already been accepted. All methods use the common CAD verifier with penetration tolerance $\max(10^{-5} D, 10^{-7})$ for assembly diameter D and the call cap in Table 25. The predeclared call budgets are $\{1.25, 1.5, 2, 2.5, 3, 4\}N$; AUC is their unweighted success mean; capped normalized call cost is the number of calls divided by $4N$ on success and 1 on failure; intervals follow the assembly-level protocol in Table 25.

CAD relation indices are (j, i, d): $B_T[j, i, d] = 1$ when predicted part j blocks part i along direction d, giving a tensor with two part axes and one direction axis. The selected assignment indexes reference geometry, and the recovered yaw lift rotates it before blocker evaluation along the six world axes. Candidate (i, d) uses its blocker column and outward projection. The transported geometry stays fixed while relations among remaining parts and current action ranks are updated. The frozen verifier uses the true part assignment and maps the world direction back through the true yaw lift. At the evaluated maximum of 300 candidates, distinct normalized average ranks differ by at least $1/[2(300 - 1)]$, so the 10^{-6} and 10^{-9} terms only resolve meet ties.

Independent 120-object CAD mechanism panels. The Algorithm and Timing contrasts in Table 2(b) use the same 120 ASAP training assemblies, disjoint from earlier CAD evaluation panels in both assembly identity and mesh content. A deterministic pseudorandom ordering selects the parts quotas in Table 25. Algorithm compares the physics oracle, PCS-ICP transport, outsideness, and SymBuild. Timing compares SymBuild with Static using its independently seeded views. Both

use the common CAD verifier and the caps and bootstrap seeds in Table 25. The paired intervals in Table 9 average the four lifts within each assembly before resampling. Alignment uses the same resampling unit over the 60 evaluation assemblies from the learned residual protocol below. Success gain is the named condition minus its comparator, and call saving is the comparator minus the named condition.

Verification and complete suffixes. The CAD verifier returns both a Boolean decision and the number of sampled sweep positions, providing the call and work measurements. In this fixed-pose model, remaining parts are reset to assembled poses and removed parts are moved away. Deleting obstacles preserves a previously feasible prescribed sweep, so skipping deleted parts in a valid removal sequence gives a complete suffix.

CAD protocol for the learned residual. The aligned versus permuted CAD intervention uses mutually disjoint sets of 60 training, 20 validation, and 60 evaluation assemblies from the same public ASAP partition. Training, validation, and evaluation use sensor seeds 601, 607, and 613, respectively, with noise of 5° applied to the process axis. Training and validation assign one deterministic C_4 lift to each assembly, while evaluation uses all four lifts. The dense 96-step verifier labels a union of candidate probes at states visited by analytic rank meet, yielding 5,022 training and 1,460 validation state-action rows. At each visited state, the probe union contains the top two candidates under the transported relation, outsideness, and rank meet, together with up to two candidates sampled without replacement from the remainder. A multilayer perceptron (MLP) with two hidden layers of width 64 is trained for 2,000 AdamW steps with seed 307, learning rate 10^{-3}, batch size 256, and validation every 100 steps; validation binary cross-entropy (BCE) selects the checkpoint at step 100. All splits use the common CAD verifier and cap $\max(3N, 24)$. Alignment intervals follow Table 25.

CAD protocol for anytime work and budget distributions. The CAD work panel uses 48 ASAP training assemblies, disjoint from earlier CAD evaluation panels in both assembly identity and mesh content. A deterministic pseudorandom order based only on assembly identity is fixed before evaluation and selects the parts quotas in Table 25. Evaluation uses the sensor seeds, lifts, and proposal cap in Table 25, with noise of five degrees applied to the process axis and the common CAD verifier. Work W is the cumulative number of sampled sweep positions checked during proposal verification, with 1–96 positions per call. At each position, assembly poses are reset and MuJoCo evaluates contact distances. Initial setup of the contact baseline and final sequence replay are excluded; a cached verification is charged its original sweep count. At work budget factor b, success requires a valid completion with $W \le \lceil 96bN \rceil$.

The work budget grid is $B = (1.25, 1.5, 2, 2.5, 3, 4)$ full sweep equivalents per part. The three predeclared interruption distributions on this ordered grid are

$$\begin{aligned} \mu_{\text{tight}} &= (.35, .25, .18, .12, .07, .03), \\ \mu_{\text{uniform}} &= (1/6, 1/6, 1/6, 1/6, 1/6, 1/6), \\ \mu_{\text{relaxed}} &= (.03, .07, .12, .18, .25, .35). \end{aligned}$$

For each distribution, $\text{AUC}_\mu = \sum_{b \in B} \mu(b) F(b)$. The CAD work AUC reported in Table 1 uses uniform weights; tight and relaxed distributions provide supplementary budget weightings. Capped normalized work cost is $W/(96 \cdot 4N)$ for a valid completion and 1 otherwise, and the CAD work "Cost reduction" entry is the paired decrease from Static to SymBuild. Intervals follow Table 25.

The supplementary diagnostic budgets in Table 13 are $\{1.00, 1.025, 1.05, 1.10, 1.15, 1.20, 1.25\}$, chosen to resolve the observed work range for valid completions 1.000–1.161 and include the original grid's starting point. This analysis reuses all $48 \times 2 \times 4 = 384$ paired SymBuild/Static conditions, preserving the verifier, rankings, sensor seeds, lifts, $4N$ cap on proposal calls, and the completion criterion $W \le \lceil 96bN \rceil$ above; final replay remains excluded from work. At each budget, the eight conditions are averaged within each assembly before 10,000 paired bootstrap draws with seed 23101. The original AUC resampling indices are shared across diagnostic budgets, reproducing its interval at $b = 1.25$.

D.3 MINI-PROGRAMS

Program generation. Each proposed Mini-Program first draws its leaf count uniformly from the requested inclusive range. A full binary expression tree is then generated recursively: for a subtree with more than one leaf, the number of leaves assigned to the left child is uniform on $\{1, \ldots, L-1\}$, and every internal operator is independently sampled uniformly from XOR, AND, and OR. The two symmetric modules receive independent literal vectors of length L whose entries are fair binary bits, and the final root operator is sampled uniformly from XOR and equality. If the two module outputs are $v_0, v_1 \in \{0, 1\}$ and the root choice is indexed by $h \in \{0, 1\}$, the generation label, which takes six possible values, is $3h + v_0 + v_1$. For a requested population of C objects, the generator fills these six labels as evenly as possible; $C = 120$ therefore gives exactly 20 programs per label. Duplicate canonical programs are rejected with the two module orders identified under terminal swap; samples for labels whose quotas are full are skipped, and the accepted population is finally shuffled. Leaf counts, tree shapes, internal operators, and literal patterns follow the sampling rules above.

Relation training and confirmation population. The frozen checkpoint uses 240 training and 60 validation objects with 3–5 leaves, disjoint from all development and confirmation objects. Training and validation use base data seed 20261222, with generator offsets $+801$ and $+802$, respectively. The relation network is trained with the settings in Table 26. Validation macro F1 selects the checkpoint and the precedence, commutativity, and orbit thresholds. The selected checkpoint and thresholds are used for confirmation. Two frozen data seeds, 20270117 and 20270129, each generate 120 ID programs with 3–5 leaves using the data seed plus 901 as the generator seed and 120 length OOD programs with 6–8 leaves using the data seed plus 902 as the generator seed, giving 240 objects per split. Object identities are pairwise disjoint across seed and split combinations and from earlier Mini-Program experiments. The deployment interface retains each direct reference independently with probability .35 in one randomly chosen anchor module, yielding the observable partial reference graph used at confirmation.

Table 26: **Mini-Program relation training.** Validation macro F1 is evaluated every 100 steps to select the checkpoint and relation thresholds. Data populations and generator seeds are specified in the preceding protocol.

Setting	Value
Network	Width 64 with three message-passing layers
Optimizer	AdamW
Training steps	800
Training seed	211
Learning rate	.002
Batch size	32
Gradient clipping	1
Validation interval	100 steps
Selected thresholds	Precedence .75, commutativity .20, orbit .10

Proposal ranks and state updates. Each program presents the same unordered event pool to every method. Events are indexed by module and expression node, with a fixed final root; candidates are unfinished events. Swapping modules exchanges corresponding event indices and fixes the root. Precedence matrices have predecessors as rows and successors as columns. Let O be the sparse observed reference graph and S the cross-module correspondence matrix, with zero diagonal and root row and column. The observed expert adjoins references propagated by $(SOS) > 0$, adds constraints from non-root events to the root, removes self edges, and takes transitive closure. The frozen relation model supplies precedence probabilities; their elementwise maximum with the aligned observed graph defines the fixed process graph. All matrices share the seeded event presentation order. At state s with remaining event mask R_s, either graph G assigns candidate a the residual predecessor mass

$$z_G(s, a) = \sum_{u \in R_s} G_{ua},$$

with lower values preferred; equal scores receive their average weak rank. The process rank r_T uses the graph combining learned and observed relations and the state rank r_X uses the transported

observed graph. SymBuild recomputes both current ranks after every accepted event. Static computes the same two ranks once at the empty prefix and subsequently only removes completed events. The exact proposal key for meet is $(\min(r_T, r_X),\ r_T + r_X,\ \max(r_T, r_X),\ \pi(a))$, where π is a deterministic presentation order fixed by the seed; the Process-only and State-only controls use r_T and r_X, respectively. The permuted control applies a fixed random permutation to all non-root event identities, conjugates only the learned precedence matrix, and keeps the observed expert aligned. For data seed d and split index $j \in \{0, 1\}$ for ID and length OOD, reference sampling uses seed $d + 1900 + 100000j$. For object index i within the group defined by the data seed and split, presentation order and planning ties use $d + 100000j + 1009i$. Every proposal is accepted exactly when the hidden ground truth precedence graph contains no unfinished predecessor of that event. The hidden directed acyclic graph (DAG) preserves a complete topological suffix after each accepted event.

Mini-Program anytime metric. The maximum budget is $2n$ verifier calls for n events, with the predeclared grid $q/n \in \{1, 1.25, 1.5, 1.75, 2\}$. Program anytime AUC is the unweighted mean of the five completion indicators on this grid, i.e., the Section 3.3 AUC definition with uniform mass on these five budgets. Capped normalized cost is $q/(2n)$ for a successful plan and 1 for an unsuccessful plan; the Program "Cost reduction" entries in Table 1 report the paired Static minus SymBuild reduction in this quantity. Intervals use 10,000 bootstrap draws paired by object (seed 24101), and the two frozen data seeds are also reported separately as replication groups.

D.4 Exact-Fill Packing

Instance generation and size shift. A packing action is $a = (i, b)$, assigning an unassigned item i of weight w_i to bin b of capacity 100. Each latent group contains exactly three items whose weights sum to 100. The largest weight w_1 is sampled uniformly from the integers 45–60, and a second weight w_2 from 12–30. The third weight is set to the residual $100 - w_1 - w_2$; sampling is repeated until the residual lies in 12–40 and at least two of the three weights differ. Groups and items are permuted, and a random permutation maps latent groups to terminal bin identities. One visible anchor per group (the heaviest item, with item index breaking weight ties) is preassigned to its transported bin, so the anchors expose a complete bijection τ from groups to bins. Among the n remaining items, $\max(2, \mathrm{round}(.25n))$ reference group labels are selected without replacement and cyclically shifted, implementing the 25% prior corruption condition. With data seeds 20271103 and 20271117, each seed contributes 120 ID instances with 4–6 bins and 120 size OOD instances with 7–8 bins. Hence ID instances contain 12–18 total items and 8–12 initially unassigned decision items, while size OOD instances contain 21–24 total items and 14–16 initially unassigned decision items.

Transported prior, proposal ranks, and state updates. Let g_i be the observed reference group label. The fixed transported structure is $i \mapsto \tau(g_i)$: the group-to-bin bijection acts on reference assignment labels while item identities remain fixed. At state s, A_s contains exactly the assignments respecting current capacities. Let $L_b(s)$ be current load and $c_i(s) = |\{b : (i, b) \in A_s\}|$. For a lexicographic key κ, $\mathrm{ordnorm}_{A_s}(\kappa)$ sorts by $(\kappa(a), a)$ and maps the zero-based position to $[0, 1]$ by division by $\max(|A_s| - 1, 1)$. The transported and residual ranks are

$$r_T(i, b) = \mathrm{ordnorm}_{A_s}(\mathbf{1}[b \neq \tau(g_i)],\ g_i,\ -w_i,\ b)\,,$$
$$r_X(i, b; s) = \mathrm{ordnorm}_{A_s}(c_i(s),\ -w_i,\ 100 - L_b(s) - w_i,\ b)\,.$$

SymBuild queries actions by $(\min(r_T, r_X), r_T, r_X, a)$. Both variants recompute r_T on current A_s; refresh also recomputes r_X after acceptance, while Static retains initial residual ranks for surviving actions. The process-only control uses r_T. The best fit control orders by $(-w_i, 100 - L_b(s) - w_i, b)$. Since $c_i(s)$ is nonincreasing in w_i, best fit and r_X induce the same ordering under the shared tie-break. Permuted transport cyclically shifts the recovered group-to-bin map.

Verification, budgets, and statistics. The verifier rejects capacity overflow and sum mismatch, then accepts only when memoized exact branch-and-bound proves that remaining weights exactly fill residual capacities. Visible anchors recover the group-to-bin correspondence while corrupted non-anchor labels affect the prior. The call grid is $q/n \in \{1, 1.125, 1.25, 1.5, 2, 3\}$ for n initially unassigned items. Packing AUC uniformly averages these budgets; capped call cost is $q/(3n)$ on success and 1 on failure, and Table 1 reports paired Static minus SymBuild reductions. Intervals

use 10,000 paired instance bootstrap draws with base seed 26101; deterministic offsets index split, contrast, metric, budget, and replication for each data seed.

D.5 IKEA Observed Process Topology

Terminal representation. We use the IKEA Video Manuals dataset from Liu et al. (2024), which provides aligned furniture parts, manuals, and substeps from real assembly videos. The resulting dataset contains 36 terminal families and 97 parsed real videos. The 267 referenced OBJ parts provide the identities and geometry used to construct the terminal representation. Terminal correspondence is computed from the exact part orbits of an unordered graph of manual incidence under automorphisms that preserve color, with witness automorphisms supplying the transport maps used for candidate closure. For each OBJ part, vertices are centered and divided by their root-mean-square (RMS) radius. The part color is determined by the normalized covariance eigenvalues in descending order, quantiles of radial distance at $0, .1, .25, .5, .75, .9, 1$, and the logarithm of the vertex count to base two; eigenvalues and radial quantiles are rounded to three decimals and the statistic for vertex count to two decimals. A mesh with zero radius receives a separate degenerate color indexed by its vertex count. The resulting color is invariant under rigid transformations and independent of pose and process order. Each official manual connection is represented by one connection node joined to two side nodes of the same color, each connected to the corresponding endpoint part. The terminal graph contains only the unordered incidence structure among parts and the part colors defined above, which are invariant under rigid transformations. Thirteen families contain nontrivial terminal orbits.

Event selection and candidate construction. The video parser admits only nonempty video IDs and substeps with a recorded start time and at least one nonempty part component. Within each video, usable events are sorted by start time and source order; consecutive exact repeats with the same action identity and start time are deduplicated. A predictive state is formed between successive usable events only when their start times differ. These deterministic parsing rules produce the reported 97 videos and 1,022 evaluable states; the latter span 35 families because one of the 36 terminal families contributes no evaluable transition. Each state is the realized video prefix up to the current substep. Its candidate set is the deduplicated union of all remaining observed actions and their variants induced by terminal transport maps, and the positive label is the actually observed next action. Of the 1,022 states, 191 states in 13 families exhibit terminal ambiguity, meaning that the next action admits a nontrivial transported alternative.

Predictors and statistical evaluation. All compared predictors use the same candidates, five folds with families held out for out-of-fold evaluation, 4,096-dimensional feature hashing, logistic regression, seed 20260802, and $C \in \{.03, .1, .3, 1, 3\}$ selected by the predeclared rule based on macro mean reciprocal rank (MRR) on the validation data. The prefix model uses candidate features and features of the realized prefix; the static orbit model adds fixed statistics of the terminal orbit; the section model refines orbit members using the prefix group in which each is first seen and its age; and the shuffled section control preserves terminal structure and prefix length while permuting which process member is assigned to the realized trajectory. Top-1 and MRR rank the true next action among candidates, with paired bootstrap intervals at the family level from 10,000 draws.

IKEA quantities in Table 2(c). "Process structure" is the Top-1 gap between the section and prefix models on all 191 states with terminal ambiguity. "Action core" uses the predeclared core subset (125 states, 9 families), in which terminal ambiguity is present, the realized prefix breaks every relevant transport, and the terminal has 8–20 part nodes. On this subset, prefix and static orbit have identical Top-1 accuracy, so the reported value is the section model's gain over either baseline. "Hybrid" applies the section model on states with terminal ambiguity and the prefix model elsewhere; its reported value is the Top-1 gap against prefix over all 1,022 evaluable states. Table 2(c) reports the paired gaps and intervals; the shuffled section control preserves terminal structure while changing which member of the terminal orbit the realized prefix selects.

D.6 Alternative Aggregation and Matched Controls

Product rule and multi-heuristic alternatives. The SayCanPay-adapted product rule is parameter-free and uses the same process and state score channels as the matched primary comparison, sorting

ascending by

$$K_{\mathrm{prod}}(a) = \big((1 + r_T(a))(1 + r_X(a)),\ r_T(a) + r_X(a),\ \max(r_T(a), r_X(a)),\ \mathrm{tie}(a)\big).$$

For CAD, the two raw channel scores are independently converted to SciPy average ranks shifted to start at zero; the first three key fields are encoded as product $+10^{-9}$ sum $+10^{-12}$ max, followed by the stable action order. Mini-Programs use the average weak ranks and deterministic presentation order defined above. Packing uses the normalized ranks above and the action tuple as its final tie-break. The MH key columns in Appendix Table 14(b) use a separate parameter-free key on the same interface

$$K_{\mathrm{MH}}(a) = \big(\min(2r_T(a), 2r_X(a) + 1),\ r_T(a) + r_X(a),\ \max(r_T(a), r_X(a)),\ \mathrm{tie}(a)\big),$$

while Appendix C.6 separately evaluates the MHA*-style 1:2 scheduler selected on the development set. For CAD, the static versions cache the initial combined score; for Mini-Programs they cache both initial weak ranks and then remove completed events; for exact-fill packing they retain the initial residual rank on surviving actions while restricting the fixed transported prior to the current candidate set. Refresh versions recompute the score or rank derived from the current state after each accepted transition. Candidate sets, verifier semantics, object splits, and interruption budgets match the primary panels using rank meet.

Matched controls. Refresh and static pairs share the instance, information channels, deterministic tie-break, proposal cap, and verifier. CAD controls using a single information channel are PCS-ICP and ASAP-style outsideness. Mini-Program process and state controls use combined learned and observed process relations and transported observed relations, respectively. Exact-fill packing controls are the process prior and best fit. Mini-Programs and exact-fill packing use permuted transport, while the CAD panel that uses a learned residual and a disjoint set of objects supplies the alignment intervention. Confirmation objects have identities disjoint from development and from one another. The comparison with the physics oracle in the CAD mechanism panel serves a diagnostic role.

D.7 Implementation Sources and Environment

External implementation provenance. The CAD source assemblies and outside-in geometric control are grounded in ASAP (Tian et al., 2024); CAD verification uses MuJoCo (Todorov et al., 2012). The product-rule comparison adapts the multiplicative scoring rule of SayCanPay (Hazra et al., 2024) using the exact joint key defined above. GRN experiments use the official GRN checkpoint and Panda OOD data (Bouhsain et al., 2025), and the population-based search comparison follows the CDGS search design of Mishra et al. (2026). Table 21 summarizes how the public SayCanPay, GRN, and PNO implementations enter matched comparisons or task-level context; the CDGS adaptation is specified separately in Appendix C.7. The implementations in the table use fixed repository revisions.

Software and hardware. Table 27 specifies the computing environment. Learned relation and GRN scoring use CUDA. Discrete verifiers, bootstrap and permutation statistics, and table aggregation run on CPU.

Table 27: **Software and hardware used for the reported experiments.**

Component	Configuration
Operating system	64-bit Windows 11
CPU and RAM	AMD Ryzen 7 8845H with 8 cores and 16 threads, 31.3 GiB RAM
GPU and driver	NVIDIA GeForce RTX 4060 Laptop with 8,188 MiB memory and driver 572.83
Python and PyTorch	Python 3.8.20 and PyTorch 2.4.1 with CUDA 12.4
Libraries	NumPy 1.24.4, SciPy 1.10.1 and Matplotlib 3.7.2