

Constraint-Aware Evolutionary Task Planning for Adaptive Modular Assembly under Execution Disturbances

Yiping Meng

y.meng@lboro.ac.uk

Loughborough University

Yiming Sun

University of Sheffield

Mounika Balina

University of Hertfordshire

Research Article

Keywords: evolutionary robotic assembly planning, assembly sequence planning, precedence constraints, constraint-preserving genetic algorithm, adaptive replanning in dynamic environments, disturbance recovery, modular assembly, preregistered evaluation

Posted Date: October 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11220230/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

Constraint-Aware Evolutionary Task Planning for Adaptive Modular Assembly under Execution Disturbances

Yiping Meng*

School of Architecture, Building and Civil Engineering, Loughborough University, UK

Yiming Sun

School of Electrical and Electronic Engineering, University of Sheffield, UK

Mounika Balina

School of Physics, Engineering and Computer Science, University of Hertfordshire, Hatfield, UK

Abstract

Robotic modular assembly requires the robot’s task order to respect assembly precedence and to be adapted when execution conditions change after work has begun. This paper presents Constraint-Aware Genetic Task Planning (CGTP), a constraint-preserving evolutionary planner whose operators keep candidate sequences within the precedence-feasible region by construction, with execution-state-aware partial replanning that freezes completed operations and re-optimizes only the remaining task graph. We evaluate CGTP under a preregistered, hash-sealed protocol on a nine-cell corpus (sizes $N = 12/16/30 \times$ three precedence regimes; ten instances per cell; fifty CRN-paired trials), against sampling and genetic baselines. On static objective quality CGTP is statistically indistinguishable from repair-based GA at $N = 12$ and $N = 16$ and modestly behind at $N = 30$ (Holm-adjusted $p = 0.018$); we report this openly. Its advantages lie elsewhere: on a canonical eight-task benchmark with exhaustive verification, the constrained replanner returns the exact best feasible suffix in all 128/128 structurally recoverable scenarios and correctly flags all 3/3 unrecoverable cases; at corpus scale it recovers a high fraction of disturbances (per class 0.881–1.000) with no budget-conditional search failures, the residual being genuine structural non-recoverability certified by an independent oracle. A preliminary simulated Franka Panda workcell audit finds all 96 end-effector poses inverse-kinematics reachable and collision-free. CGTP thus trades a small, size-dependent static-objective premium for precedence-feasibility preservation and reliable disturbance replanning; robot-level execution feasibility is a subsequent extension.

Keywords: evolutionary robotic assembly planning; assembly sequence planning; precedence constraints; constraint-preserving genetic algorithm; adaptive replanning in dynamic environments; disturbance recovery; modular assembly; preregistered evaluation.

1 Introduction

Automatic assembly planning sits between symbolic decision-making and physical robot execution. Before a robot moves, the system must decide which component to assemble next, whether the proposed order respects technological dependencies, and whether changes in the environment require the remaining plan to be revised. These questions are especially important in modular assembly, where components are connected through explicit precedence relationships and where an early sequencing error can make later operations impossible.

*Corresponding author: y.meng@lboro.ac.uk

Assembly planning has been represented using directed graphs, AND/OR structures and task-sequence graphs, while optimization methods have included heuristic search, genetic algorithms (GAs), reinforcement learning and graph-learning methods [1–8]. In parallel, motion-planning research provides mature tools for reachability, inverse kinematics and collision-free movement [9, 10]. These streams solve important parts of the problem but do not always answer the same question. A graph can describe which orders are legal without prescribing how an optimizer should search them; a GA can explore permutations without guaranteeing that its intermediate candidates respect assembly dependencies; and a precedence-valid plan is a task-level object that still requires separate robot-level checking before physical execution.

The focus of this study is the *robotic assembly task-planning* layer: how explicit precedence structure should shape an evolutionary search over the order in which a robot assembles modular components, and how that order should be repaired after an execution disturbance. The decision variable is the robot’s assembly task sequence under precedence and execution-state constraints—not a generic scheduling index—and the output of disturbance replanning is the precedence-consistent remaining task sequence that a downstream robot execution layer would carry out. Robot-level motion feasibility—inverse kinematics, reachability, collision and sequential executability—is a distinct layer that we treat as a subsequent extension rather than as a result of the present study; the present contribution is the evolutionary task-level planner that supplies that layer with feasible, disturbance-adapted sequences.

Three connected gaps motivate the work. First, the measurable effect of using explicit precedence information to guide an evolutionary search is not always isolated from the effect of repair or of generic optimization: conventional permutation search can spend effort outside the feasible precedence region, and repair can restore feasibility but changes the search dynamics. Second, disturbance handling in assembly should preserve operations already completed and reconsider only the unfinished task graph—partial replanning from an immutable execution prefix—rather than treating every change as a fresh planning problem. Third, there is limited controlled evidence on the *trade-off between feasibility-preserving evolutionary operators and repair-based search* across problem size and constraint regime.

Satisfying precedence is necessary but not sufficient for good optimization; feasibility and objective quality are distinct properties, and a method can guarantee the first without winning the second. The overall research question is therefore: *how does explicit assembly-precedence knowledge, used to constrain an evolutionary planner by construction, compare with repair-based and unguided search in static objective quality, and how reliably can such a planner recover feasible sequences after execution disturbances?*

- **RQ1.** How does constraint-preserving evolutionary optimization compare with unguided, penalty-based and repair-based approaches in static objective quality and in the feasibility of its evolutionary search, across problem size and constraint regime?
- **RQ2.** Can execution-state-aware partial replanning recover feasible sequences after disturbances while preserving completed tasks, and can it distinguish structural non-recoverability from search failure?

The study makes three scientific contributions and one evaluation statement.

- **C1 — A task-level precedence-feasibility-preserving evolutionary planning and replanning architecture.** CGTP constrains initialization and variation to the precedence-feasible region by construction, so that the evolutionary evaluation stream remains precedence-valid without routine repair, and it replans the remaining task graph from an immutable execution prefix.
- **C2 — Exact-verified disturbance replanning on the canonical benchmark.** On an eight-task case small enough for exhaustive enumeration, the constrained replanner’s output is checked against the exact best feasible suffix, providing a correctness anchor at verifiable scale.

- **C3 — Corpus-scale recovery evaluation with explicit attribution.** Across a nine-cell corpus, disturbance recovery is quantified and its complement is separated into structural non-recoverability (certified by an algorithm-independent oracle) versus budget-conditional search failure.

Evaluation statement. The above is assessed using a preregistered, seed-matched, instance-level statistical protocol with frozen benchmarks, budgets and analysis rules.

An honest scope statement frames the results: *on raw static objective value, CGTP is competitive with but not superior to repair-based GA; the framework’s main value lies in structural feasibility preservation and disturbance recovery.*

2 Related Work

2.1 Robotic assembly planning

Assembly sequence planning has a long combinatorial and geometric tradition, surveyed comprehensively by [11]. Graph representations express assembly dependencies naturally. Classical AND/OR graph formulations showed how alternative decompositions and legal operation orders can be represented systematically [1], and later work extended graph reasoning toward automatic extraction of assembly knowledge from structured instructions [2]. These methods make assembly relationships explicit and machine-readable; beyond representation, assembly constraints extracted from existing products can be recognized and transferred to new product variants to reduce the sequence-planning search space [8]. The question for the present study is not whether a graph can represent an assembly, but how that graph should influence optimization: CGTP uses a directed precedence task graph as an active constraint on the evolutionary search, which lets the experiments measure how much search effort remains within the feasible region.

2.2 Evolutionary and learning-based sequence planning

Genetic algorithms have been applied to assembly sequencing because permutation representations map naturally to ordered task lists [3]. The difficulty is that conventional crossover and mutation do not preserve precedence, so in a sparse feasible space an unguided GA may evaluate many chromosomes that cannot represent a legal order. Constraint handling in evolutionary computation is itself a well-studied area [12, 13]: repair-based constraint handling—restoring feasibility after variation—forms one major family [14] and is a strong baseline, while the alternative is to design operators that preserve feasibility by construction, the route CGTP takes. Learning-based methods—graph neural networks and deep reinforcement learning—have been used to rank assembly decisions and incorporate feasibility [4–7]; their emphasis is typically policy or representation learning. The present work is deliberately narrower: it asks what happens when explicit precedence structure is built into the operators of a transparent evolutionary optimizer, and compares that with unguided, penalty-based and repair-based alternatives under one frozen benchmark. The central literature contrast is thus *feasible-by-construction operators versus generate-then-repair constraint handling*, evaluated rather than assumed.

2.3 Adaptive planning and replanning

Online replanning is well established for path changes and dynamic obstacles, most notably through incremental heuristic search that reuses previous computation when the environment changes—D* [15], D* Lite [16] and Anytime Dynamic A* [17]—and for resolving conflicts among multiple agents, for example conflict-based search [18]. These methods repair a path or a set of paths in a metric space when costs or obstacles change. Assembly task replanning introduces a different state constraint: some

operations may already be physically completed, so treating them as unexecuted decisions is unrealistic. A useful assembly replanner retains the executed prefix and reconstructs only the remaining problem—closer in spirit to receding-horizon reuse than to full re-solution. In manufacturing more broadly, handling disruptions online is the subject of dynamic scheduling and rescheduling [19], though that literature typically re-optimizes job or resource timing rather than a precedence-constrained assembly task order under geometric disturbances. The disturbance model used here covers geometric, temporal and execution-related changes—workspace obstruction, component displacement, execution delay, path-length increase and recoverable task failure—as controlled tests of whether the planner distinguishes disturbances that require resequencing from those for which the current remaining order can be retained.

2.4 Task-level versus robot-level feasibility (scope)

Sampling-based motion planners and their asymptotically optimal variants evaluate robot trajectories in continuous configuration spaces [9, 10, 20], and assembly studies have incorporated inverse kinematics, accessibility and collision constraints, including iteratively refined robot-feasibility checking within sequence planning [6, 21]. These establish that symbolic task planning cannot be equated with physical execution: a precedence-valid order is not by itself proof that a manipulator can execute every transition. The tightest integration of the two levels is task-and-motion planning (TAMP), which interleaves symbolic action selection with geometric motion feasibility so that discrete choices are validated against continuous constraints [22–25]; TAMP frameworks are powerful but couple the layers into a single, computationally heavy search. The present paper is deliberately scoped to the *task level* and keeps the two layers separate: it studies precedence-feasible evolutionary planning and disturbance recovery, and treats robot-level execution feasibility as a distinct layer and subsequent extension (§5.4); the preliminary reachability audit of §4.5 is a first, lightweight step toward that layer rather than a TAMP-style coupled solver. The gap addressed here is the lack of a controlled quantitative comparison of precedence-feasible-by-construction evolutionary search against repair-based search, together with partial task-level recovery after disturbances, under a preregistered protocol.

3 Methodology

Figure 1 gives an overview of the framework: constraint-preserving evolutionary static planning, followed by disturbance-triggered partial replanning that freezes the executed prefix, consults a structural recoverability oracle, and (where recoverable) re-optimizes the remaining suffix under CGTP, classifying each episode by a frozen four-way outcome taxonomy.

3.1 Benchmark system and corpus

Two benchmark tiers are used. A **canonical eight-task instance (N8-CANONICAL)** retains the established benchmark (tasks Base, A, B, C, D, E, F, G; durations and planar target positions in Table 1; eight precedence edges Base→A, Base→B, A→C, B→D, C→E, D→F, E→G, F→G; the precedence DAG is shown in Figure 2). It is the only $N = 8$ instance and is small enough for exhaustive verification: of $8! = 40,320$ permutations, only **20 (0.0496%)** satisfy all precedence constraints, so 99.95% of arbitrary orders are infeasible before any geometric consideration. This sparsity is the primary motivation for precedence-guided search.

A **frozen corpus** provides statistical breadth: problem sizes $N \in \{12, 16, 30\}$ crossed with three precedence regimes (Sparse, Moderate, Dense) defined by the feasible-ordering structure of the precedence poset (normalized ordering entropy, with the feasible-ordering ratio as a companion metric), giving nine cells with **ten instances per cell (90 instances)**. Instances are generated once under a frozen seed schedule and **shared identically across all algorithms**; every algorithm sees the same instances and,

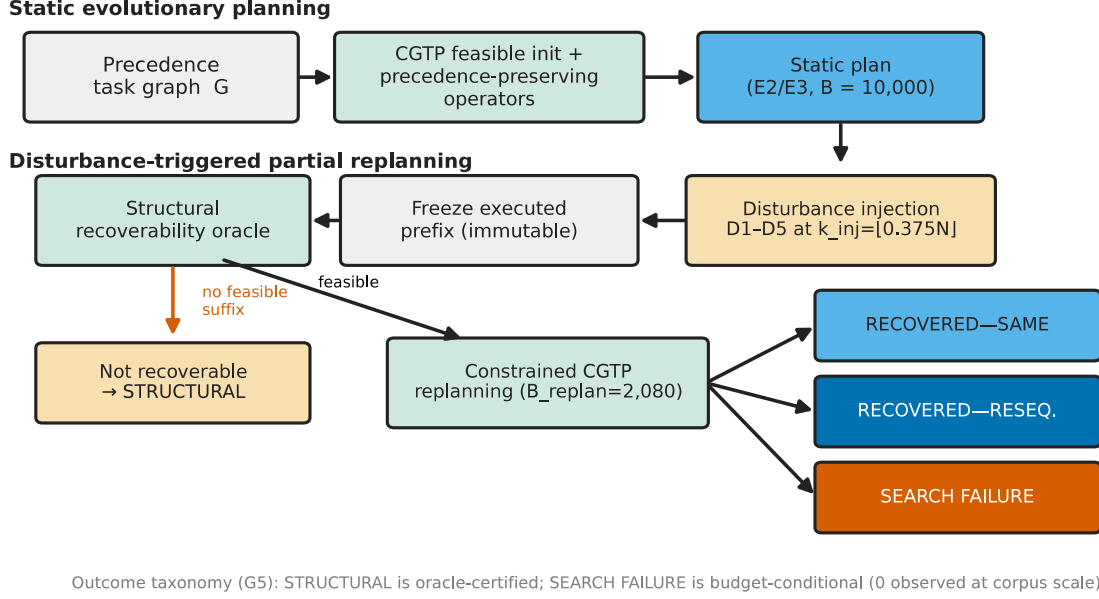


Figure 1: Framework overview: constraint-preserving evolutionary static planning, then disturbance-triggered partial replanning (freeze executed prefix → structural recoverability oracle → constrained CGTP replanning → four-way outcome taxonomy).

within an instance, the same trial seeds (Table 2).

3.2 Objective

Candidate orders are scored by the frozen composite objective. For an order σ , travel is the summed Euclidean distance between consecutive target positions and completion is the sum of task completion (flow) times, each normalized to $[0, 1]$ by predetermined per-instance bounds: the travel lower bound T_{LB} is the Euclidean minimum-spanning-tree weight over the target positions and the travel upper bound T_{UB} is the sum of the $N - 1$ largest pairwise target distances, while the completion bounds are the exact shortest- and longest-processing-time (SPT/LPT) orders. These predetermined bounds place both normalized components—and hence J —in $[0, 1]$ for every feasible order. The objective is

$$J(\sigma) = 0.60 \cdot \text{TravelNorm}(\sigma) + 0.40 \cdot \text{CompletionNorm}(\sigma), \quad (1)$$

minimized over feasible orders. The weights (0.60/0.40) and normalization are fixed before any comparative run. There is **no precedence-gap term**: precedence is a hard feasibility constraint, not a rewarded objective component. On N8-CANONICAL the exact constrained optimum is $\sigma^* = \text{Base-A-C-E-B-D-F-G}$ with $J^* = 0.159340$ (travel 2.5724 m, TravelNorm 0.0989; completion 61, CompletionNorm 0.2500), rank 1 of the 20 feasible orders.

3.3 Algorithms

Five methods span increasing constraint awareness, evaluated under a common budget (Table 3). A4 and A5 differ in *where* feasibility is enforced. A4 applies ordinary crossover and mutation and then repairs the result. A5 preserves precedence feasibility structurally, through constrained initialization and precedence-preserving operators, so that repair is not part of the routine search path.

Table 1: Canonical benchmark (N8-CANONICAL) tasks: durations and planar target positions. Travel is computed as the Euclidean distance between consecutive target positions under the composite objective. Edges: Base→A, Base→B, A→C, B→D, C→E, D→F, E→G, F→G. Feasible orders: 20/40,320 (0.0496%). Exact optimum $\sigma^* = \text{Base-A-C-E-B-D-F-G}$, $J^* = 0.159340$.

ID	Task	Duration (s)	Target x (m)	Target y (m)
1	Base	0	0.45	0.25
2	A	2	0.75	0.20
3	B	3	0.55	0.55
4	C	1	1.00	0.35
5	D	4	0.80	0.75
6	E	2	1.25	0.55
7	F	3	1.05	0.95
8	G	1	1.35	0.85

Table 2: Corpus design. Problem sizes $N \in \{12, 16, 30\}$ crossed with three precedence regimes, ten instances per cell, plus the canonical instance; all algorithms share instances and per-trial seeds.

Factor	Levels
Problem size N	12, 16, 30
Precedence regime	Sparse, Moderate, Dense
Cells	$3 \times 3 = 9$
Instances per cell	10 (90 total) + N8-CANONICAL
Trials per instance	50 (CRN-paired)
Static budget B	10,000 evaluations
Replan budget B_{replan}	2,080 evaluations
Injection point	$k_{\text{inj}} = \lfloor 0.375 N \rfloor$

3.4 CGTP operators

CGTP uses randomized feasible initialization (uniform choice from the currently available tasks, ascending-ID enumeration), a precedence-preserving crossover, and a feasible-window relocation mutation. A post-variation precedence safeguard checks every candidate before evaluation; if a precedence violation is detected, the event is logged and deterministic repair is available as a defensive fallback. Because the CGTP operators are precedence-preserving, no precedence-repair invocation occurred in the formal campaign. During disturbed replanning, candidates that violate active exclusion constraints are rejected under the separately frozen replanning safeguard, which retains the pre-variation parent. Figure 7 contrasts this feasible-by-construction operator flow with the generate-then-repair flow of GA+Repair (A4).

3.5 Fairness and budgets

Comparisons use matched evaluation budgets, common random numbers (the same instance–trial seed drives every algorithm, so trials are paired by construction), shared instances, a common seed structure, and the same objective evaluator. The **static optimization budget** is $B = 10,000$ objective evaluations per run (a generational schedule of 100 post-initial generations). The **replanning budget** is $B_{\text{replan}} = 2,080$ per replanning round.

Canonical eight-task precedence graph (20 of 40,320 orders feasible; 0.0496%)

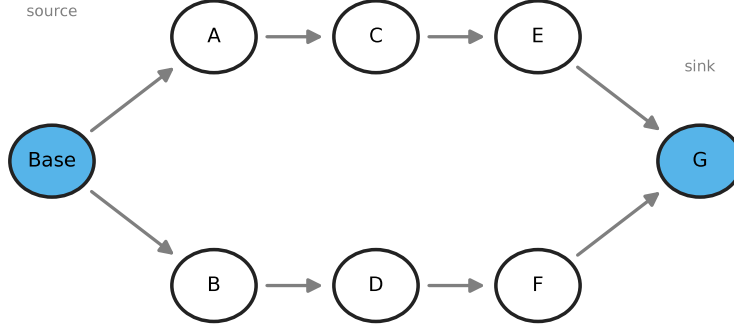


Figure 2: Directed precedence graph of the canonical eight-task benchmark. Of $8! = 40,320$ permutations only 20 (0.0496%) satisfy all precedence constraints, which motivates precedence-guided search.

Table 3: Five methods spanning increasing constraint awareness, evaluated under a common budget.

Label	Method	Constraint handling
A1	Uniform-permutation sampling	none (penalty-graded)
A2	Constrained-init sampling	feasible-by-construction sampling
A3	Penalty GA	infeasibility penalized in fitness
A4	GA + Repair	ordinary variation, then repair to feasibility
A5	CGTP	precedence feasibility preserved by construction

3.6 Disturbance protocol and partial replanning

Five disturbance classes are used, each at Low, Moderate and High severity: **D1** workspace obstruction (a positional exclusion window), **D2** component displacement (a target position shift), **D3** execution delay (a duration increase), **D4** path-length increase (a directed travel-cost increase), and **D5** recoverable task failure (the plan-head task excluded for a window). Disturbances are injected after the task at position $k_{inj} = \lfloor 0.375N \rfloor$ of the executed plan (on N8-CANONICAL, after Base-A-C, leaving a five-task remaining problem). Partial replanning freezes the completed prefix and re-optimizes only the remaining suffix under the disturbed instance; outcomes are classified by a frozen taxonomy: **recovered–same order**, **recovered–resequenced**, **not recoverable–structural** (no feasible suffix exists, certified by an algorithm-independent structural recoverability oracle), and **search failure–budget-conditional** (a feasible suffix exists but was not found within B_{replan}).

3.7 Exact verification

On N8-CANONICAL the remaining problem is small enough for exhaustive enumeration, which serves as an *independent* ground-truth verifier (not part of the planner): the exact best feasible suffix and the exact recoverability verdict are computed by brute force and compared with the planner’s output. This canonical verification uses a frozen 168-scenario battery (E4a) that instantiates the five classes on the five unfinished tasks at the three severities; because D2 displaces a target in a continuous plane, it additionally sweeps eight fixed displacement directions (so D2 contributes $5 \times 8 \times 3 = 120$ scenarios and D1/D3/D4 contribute $5 \times 3 = 15$ each, with three D5 plan-head cases), and displacements that leave the workspace

bounding box are labelled geometrically inadmissible and excluded from the denominator.

3.8 Statistics

Analysis follows the frozen statistical plan. The **instance is the inferential unit**: cell-level evidence is the ten paired instance-level summaries, and run-level counts are never used as test degrees of freedom. For static quality, per instance the paired difference $d_i = \text{median over its 50 trials of } [J(A) - J(A')]$; the cell estimand is $\Delta_c = \text{median}_i d_i$, tested by an exact Wilcoxon signed-rank test with Holm correction across cells, reported with 95% instance-cluster bootstrap confidence intervals ($B_{\text{boot}} = 10,000$) and rank-biserial effect sizes. For recovery, the confirmatory test is a one-sided exact Wilcoxon on the ten instance-level recovery proportions against 0.5 at Moderate severity, Holm-corrected across disturbance classes; recovery rates are reported as estimation with cluster-bootstrap CIs. Exploratory analyses (e.g., size trends) are labelled as such.

3.9 Reproducibility

The instrument is frozen and hash-sealed: fixed protocol specifications (G1–G9), a fixed seed schedule, a fixed corpus, mandatory checkpointing with deterministic resume, hash-verified execution, and a preregistered analysis plan whose validity gates are checked before results are unblinded. Full provenance (environment continuity, substrate qualification, and the analysis audit) is given in supplementary material.

4 Results

4.1 Static plan quality and scaling

The estimated objective differences place A4 and A5 well below the sampling baselines across most cells (median $\Delta J \approx -0.76$ versus uniform sampling A1 in all nine cells; A5/A4 below constrained-init sampling A2 in eight of nine). Because the preregistered P2 multiplicity family is ambiguous, these contrasts are interpreted descriptively rather than as confirmatory superiority tests; they indicate that precedence feasibility alone (A2) does not deliver good objective quality.

Within the strong search tier, **CGTP does not outperform GA+Repair**. The paired instance-level difference $\Delta_c = \text{median}_i \text{median}_t [J(A5) - J(A4)]$ is ≥ 0 in all nine cells ($\Delta > 0$ in five, exact ties in four at $N = 12/16$). At $N = 12$ and $N = 16$ the two methods are statistically indistinguishable (effect ≈ 0 ; no Holm-significant cell). **At $N = 30$, GA+Repair is significantly better in all three regime cells** ($\Delta = +0.029$ Sparse, $+0.023$ Moderate, $+0.007$ Dense; Holm-adjusted $p = 0.018$; rank-biserial $+1.0$, i.e. A5 is worse in every one of the ten instances). Comparison with the penalty GA (A3) is a near-tie: the median paired difference is ≈ -0.0002 for the A5–A3 contrast and -0.0035 for the A4–A3 contrast (both marginally in favour of the strong search methods over the penalty GA; no Holm-significant cell). Because the confirmatory family for the sampling/penalty baseline contrasts (P2) is specified ambiguously in the analysis plan, those contrasts are reported as estimation (effect sizes with CIs) rather than as confirmatory tests.

The size trend is descriptive (Figure 6): the A5–A4 median gap grows with N (≈ 0 at $N = 12$, $+0.0002$ at $N = 16$, $+0.020$ at $N = 30$). Preferred statement: *among the strong search methods, static solution quality is closely clustered at $N = 12$ and $N = 16$, while repair-based GA obtains a small but statistically significant advantage at $N = 30$* . Per-cell Δ_c , CIs and effect sizes are in Table 4; the forest plot is Figure 3.

Table 4: P1 — static quality, A5 (CGTP) vs A4 (GA+Repair), per cell. Δ_c is the median paired J difference ($> 0 \Rightarrow$ CGTP worse); 95% cluster-bootstrap CI; Holm-adjusted p ; rank-biserial effect size. $\Delta_c \geq 0$ in all nine cells; Holm-significant at all three $N = 30$ cells.

N	Regime	Δ_c	95% CI	p_{Holm}	r_{rb}	Sig.
12	Sparse	0.0000	[0.000, 0.013]	0.547	+1.00	
12	Moderate	0.0000	[−0.005, 0.000]	1.000	−0.40	
12	Dense	0.0000	[0.000, 0.000]	1.000	0.00	
16	Sparse	0.0017	[0.000, 0.009]	0.547	+0.67	
16	Moderate	0.0045	[−0.001, 0.011]	0.469	+0.72	
16	Dense	0.0000	[0.000, 0.007]	1.000	+1.00	
30	Sparse	0.0292	[0.020, 0.040]	0.018	+1.00	*
30	Moderate	0.0230	[0.018, 0.037]	0.018	+1.00	*
30	Dense	0.0068	[0.003, 0.012]	0.018	+1.00	*

Table 5: Canonical exact-verification battery (E4a) on N8-CANONICAL (131 admissible = 128 recoverable + 3 structural). The all-100% accuracy bar chart is in supplementary material.

Scenario category	Count	Planner outcome
Admissible & recoverable	128	exact best suffix: 128/128 (100%)
Structurally unrecoverable	3	correctly flagged: 3/3 (100%)
Geometrically inadmissible	37	excluded by protocol
Total scenarios	168	

4.2 Exact-verified replanning correctness

On the canonical benchmark, CGTP’s constrained replanner is exact. The frozen 168-scenario disturbance battery contains **131 geometrically admissible** scenarios (**37** geometrically inadmissible displacements are excluded from the denominator by protocol). Of the 131 admissible scenarios, **128 are structurally recoverable** and **3 are structurally unrecoverable**. On *all* 128 recoverable scenarios the constrained replanner returned the exact best feasible suffix confirmed by exhaustive enumeration (**128/128, 100%**), and **all 3/3** structurally unrecoverable cases were correctly identified as not recoverable (Table 5). The recoverable set spans all five disturbance classes (D1: 12, D2: 83, D3: 15, D4: 15, D5: 3). This provides an exact correctness anchor for the replanning mechanism at a scale where exhaustive verification is possible. It is a verification result, not a comparison against a competitor.

4.3 Corpus-scale disturbance recovery

Across the 90-instance corpus (67,500 replanning runs), recovery (recovered–same + recovered–resequenced) is high (Figure 4): per class **D1** ≈ 0.978 , **D2** = 1.000, **D3** = 1.000, **D4** = 1.000, **D5** ≈ 0.881 . The confirmatory recovery-floor test (median instance recovery > 0.5 at Moderate severity) is Holm-significant in **43 of 45** cell $\times$ class slices; the two non-significant tests occur for D5 in the $N = 12$ and $N = 16$ Dense cells, whose point estimates remain high but for which the preregistered instance-level tests do not reject the 0.5 recovery-floor null at the corrected threshold (the power limitation is discussed in §5.4).

Two properties matter for interpretation. First, there were **zero budget-conditional search failures**: whenever the oracle certified recoverability, CGTP found an admissible suffix within $B_{\text{replan}} = 2,080$, so the residual non-recovery ($\approx 2.8\%$ overall) is **structural non-recoverability** certified by the algorithm-independent oracle, not planner failure. Second, recovery is roughly evenly split between **recovered–same order** (≈ 0.454) and **recovered–resequenced** (≈ 0.518) (Figure 5): recovery is frequently genuine

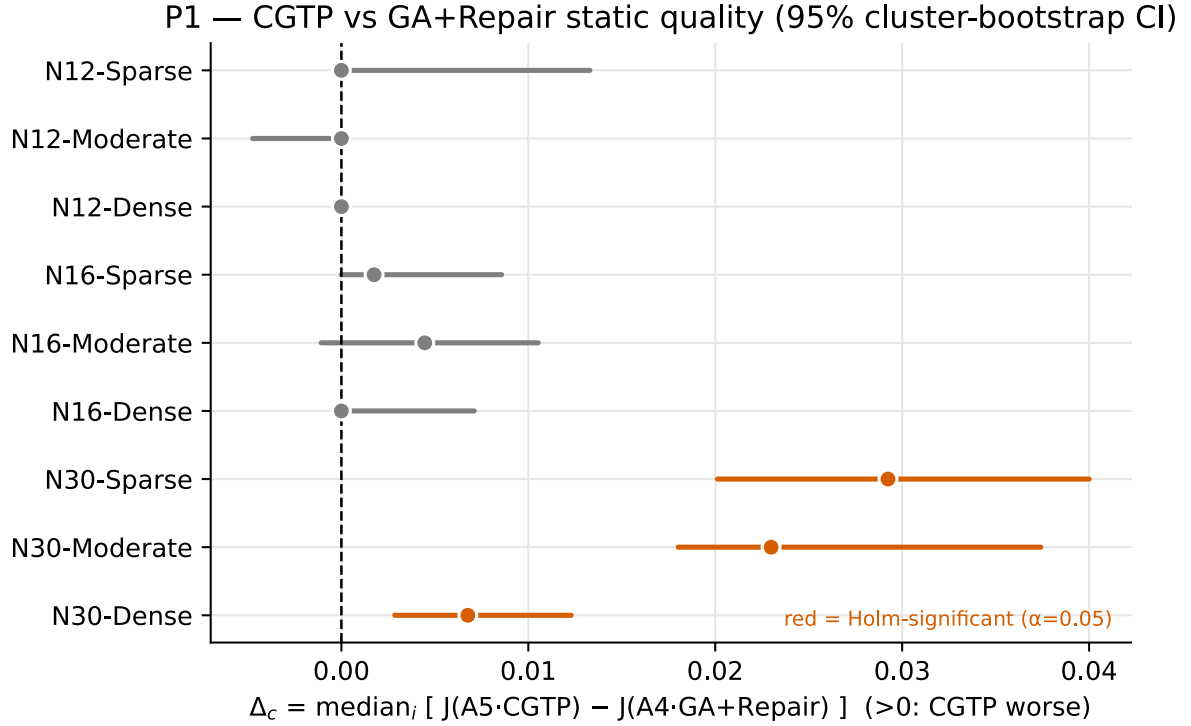


Figure 3: P1 forest plot — Δ_c (A5—A4) static-quality difference with 95% confidence intervals across the nine cells. Positive values indicate CGTP is worse; the three $N = 30$ cells are Holm-significant.

resequencing rather than trivial continuation of the nominal order, and the resequenced share rises with severity. Because E4b evaluates CGTP only, these are capability and attribution results, not a comparative recovery claim. Observed aggregate recovery was somewhat higher at larger N in this corpus (0.962, 0.968, 0.985 for $N = 12, 16, 30$); this is a descriptive observation about the corpus, not a general scaling law. The recovery rate by class and severity is in Table 6; the four-way outcome composition is Figure 5 (full per-cell composition in supplementary material).

Table 6: E4b corpus recovery decomposition per disturbance class $\times$ severity $\times N$: recovered–same, recovered–resequenced, not-recoverable–structural, and search-failure rates (means over instance-level trials). Search failure is 0 throughout; recovery = same + reseq.

Class	Sev. (N)	Recovery	Same	Reseq.	Not-rec.	Search-fail
D1	Low (12)	0.992	0.801	0.191	0.008	0.000
	Low (16)	0.989	0.711	0.279	0.011	0.000
	Low (30)	1.000	0.627	0.373	0.000	0.000
	Moderate (12)	0.965	0.697	0.268	0.035	0.000
	Moderate (16)	0.983	0.638	0.345	0.017	0.000
	Moderate (30)	0.994	0.585	0.409	0.006	0.000
	High (12)	0.946	0.588	0.358	0.054	0.000
	High (16)	0.953	0.559	0.395	0.047	0.000
	High (30)	0.983	0.558	0.425	0.017	0.000
D2	Low (12)	1.000	0.739	0.261	0.000	0.000
	Low (16)	1.000	0.612	0.388	0.000	0.000
	Low (30)	1.000	0.617	0.383	0.000	0.000
	Moderate (12)	1.000	0.549	0.451	0.000	0.000
	Moderate (16)	1.000	0.455	0.545	0.000	0.000
	Moderate (30)	1.000	0.552	0.448	0.000	0.000
	High (12)	1.000	0.399	0.601	0.000	0.000
	High (16)	1.000	0.301	0.699	0.000	0.000
	High (30)	1.000	0.462	0.538	0.000	0.000
D3	Low (12)	1.000	0.673	0.327	0.000	0.000
	Low (16)	1.000	0.637	0.363	0.000	0.000
	Low (30)	1.000	0.610	0.390	0.000	0.000
	Moderate (12)	1.000	0.481	0.519	0.000	0.000
	Moderate (16)	1.000	0.503	0.497	0.000	0.000
	Moderate (30)	1.000	0.542	0.458	0.000	0.000
	High (12)	1.000	0.365	0.635	0.000	0.000
	High (16)	1.000	0.352	0.648	0.000	0.000
	High (30)	1.000	0.421	0.579	0.000	0.000
D4	Low (12)	1.000	0.680	0.320	0.000	0.000
	Low (16)	1.000	0.677	0.323	0.000	0.000
	Low (30)	1.000	0.624	0.376	0.000	0.000
	Moderate (12)	1.000	0.619	0.381	0.000	0.000
	Moderate (16)	1.000	0.607	0.393	0.000	0.000
	Moderate (30)	1.000	0.605	0.395	0.000	0.000
	High (12)	1.000	0.540	0.460	0.000	0.000
	High (16)	1.000	0.501	0.499	0.000	0.000
	High (30)	1.000	0.553	0.447	0.000	0.000
D5	Low (12)	0.933	0.000	0.933	0.067	0.000
	Low (16)	0.898	0.000	0.898	0.102	0.000
	Low (30)	1.000	0.000	1.000	0.000	0.000
	Moderate (12)	0.833	0.000	0.833	0.167	0.000
	Moderate (16)	0.891	0.000	0.891	0.109	0.000
	Moderate (30)	0.930	0.000	0.930	0.070	0.000
	High (12)	0.767	0.000	0.767	0.233	0.000
	High (16)	0.808	0.000	0.808	0.192	0.000
	High (30)	0.867	0.000	0.867	0.133	0.000

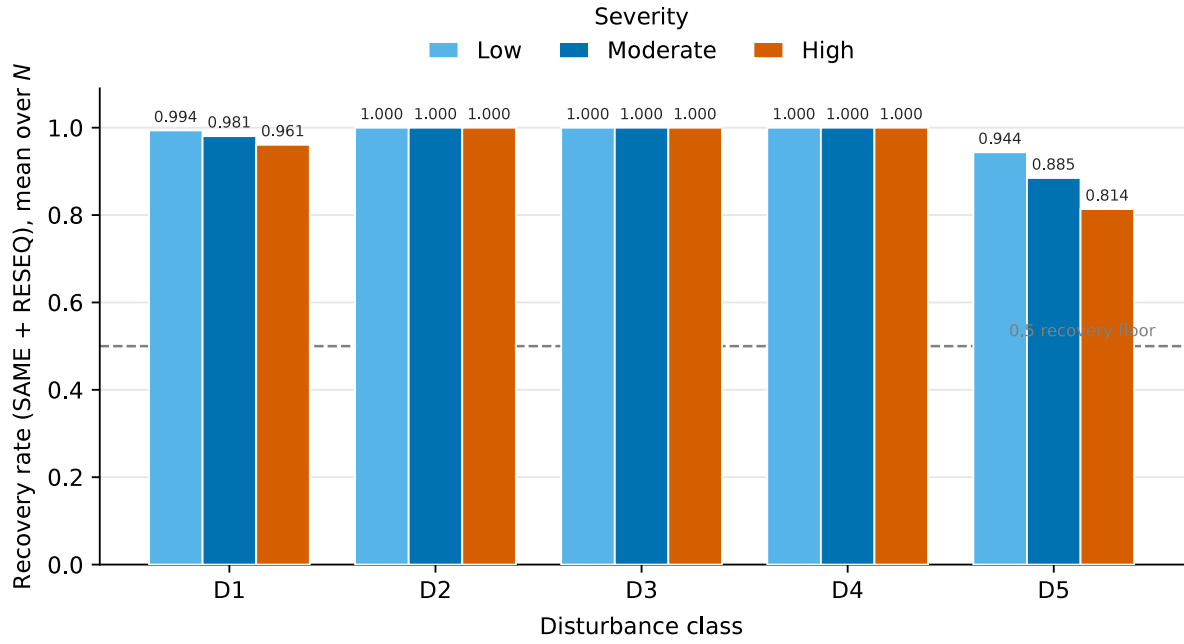


Figure 4: E4b recovery rate by disturbance class and severity across the 90-instance corpus.

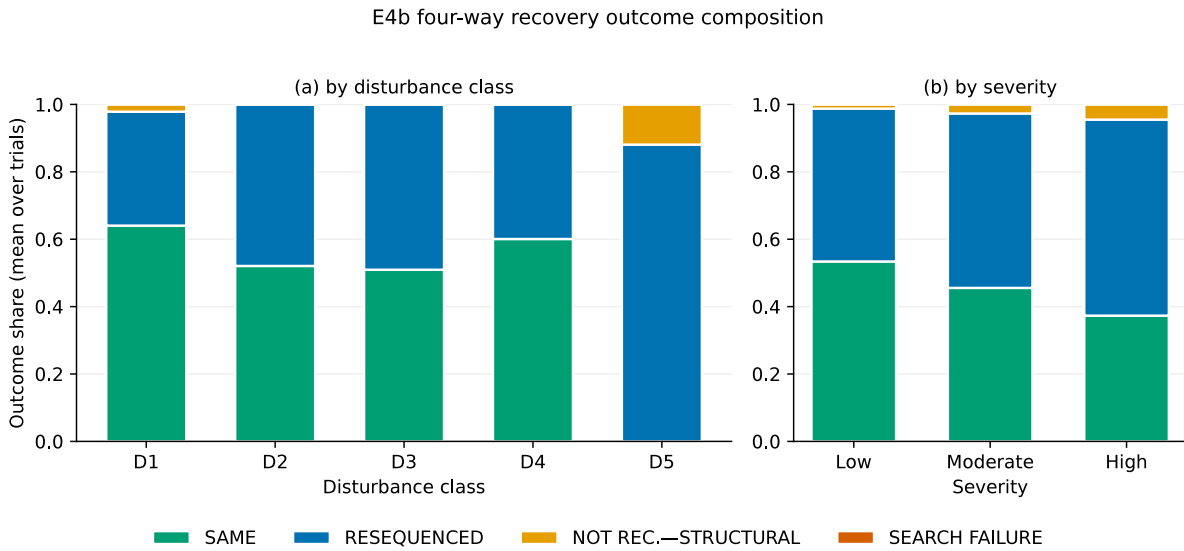


Figure 5: E4b four-way recovery outcome composition (SAME / RESEQUENCED / NOT-RECOVERABLE-STRUCTURAL / SEARCH-FAILURE), by disturbance class and by severity; SEARCH-FAILURE = 0 throughout.

4.4 Scaling behaviour and the feasibility–efficiency trade-off

The static A5–A4 gap increases with problem size (Figure 6), from an exact tie at $N = 12$ to a significant advantage for repair-based GA at $N = 30$. Read descriptively, this indicates a trade-off pattern: constraining variation to the feasible region by construction costs little at small and medium size but incurs a small objective premium at the largest size tested. The experiments establish this performance pattern; §5.1 discusses a candidate explanation without claiming a proven causal mechanism. Full robot-level execution feasibility of the planned sequences (the complete E5/E6/E6b programme) is not evaluated here and is a subsequent extension (§5.4); a preliminary pose-level reachability audit is reported in §4.5.

4.5 Preliminary simulated reachability audit (E5-lite)

To position the task-level plans within a simulated robotic workcell—addressing the “physical and simulated environments” dimension while remaining within the task-level scope of this study—we ran a lightweight post-validation reachability audit. This is explicitly a *preliminary* audit and is *not* a robot-level feasibility claim. The eight canonical target positions (Table 1) were placed on a planar board in front of a standard 7-DOF manipulator (Franka Emika Panda, simulated in PyBullet) via a fixed spatial transform (a similarity: uniform scale $s = 0.40$ plus translation, board centre 0.45 m in front of the base, board top 0.15 m). Inverse kinematics was solved with PyBullet’s damped-least-squares solver (`calculateInverseKinematics`) from a fixed nominal “ready” rest configuration, up to 600 iterations at a 10^{-5} residual threshold, with a fixed top-down tool orientation; a pose counts as *reachable* when the returned configuration satisfies all joint limits and its forward-kinematics position error is below 8 mm (a pre-specified acceptance tolerance; the realised maximum was 2.0 mm). Collision detection (PyBullet discrete queries) was evaluated at each target pose against the board, floor, the arm itself (self-collision), and, for D1, a keep-out box in the exclusion window; the audit checks the target poses, not the intervening approach segments, which belong to the robot-level layer.

All eight poses of the static optimum σ^* were IK-reachable within joint limits (maximum forward-kinematics error 2.0 mm) and collision-free (Table 7, Figure 8). The coverage argument depends on whether a disturbance moves a target. For the *position-preserving* classes—D1 (workspace obstruction), D3 (execution delay), D4 (path-length increase) and D5 (recoverable task failure)—every replanned suffix is a permutation of the unchanged canonical task set, so its end-effector pose set is a subset of the eight canonical poses; reachability transfers directly, and collision-freedom transfers for poses outside the active exclusion window. D2 (component displacement) is different: it relocates a target to a new position that is not among the eight, so we audit the displaced poses explicitly. Every admissible D2 displaced position (all 79 distinct positions arising from the five unfinished targets $\times$ eight fixed directions $\times$ three magnitudes that fall inside the protocol bounding box—a superset of the 83 recoverable D2 scenarios) was IK-reachable and collision-free. In total 96/96 audited poses passed (the eight σ^* poses, four remaining poses under an illustrative D1 keep-out, five under a D5 suffix, and the 79 D2 displaced positions; Table 7). We note that under the fixed top-down approach assumed in this audit, where two target cells are physically adjacent a top-down grasp at one cell can intersect a keep-out placed on the other; this is a limitation of the fixed approach angle, not of reachability, and is precisely the coupling that motivates task-level resequencing and, at the robot level, an approach-angle adjustment (the single D1 keep-out audited here is one illustrative instance, not an enumeration of all D1 windows). This audit establishes that the planned task sequences correspond to attainable, collision-free end-effector poses for a representative arm in simulation; it does not establish full motion-level executability, which—together with the closed-loop execution-time-feedback study (E5/E6)—remains future work (§5.4).

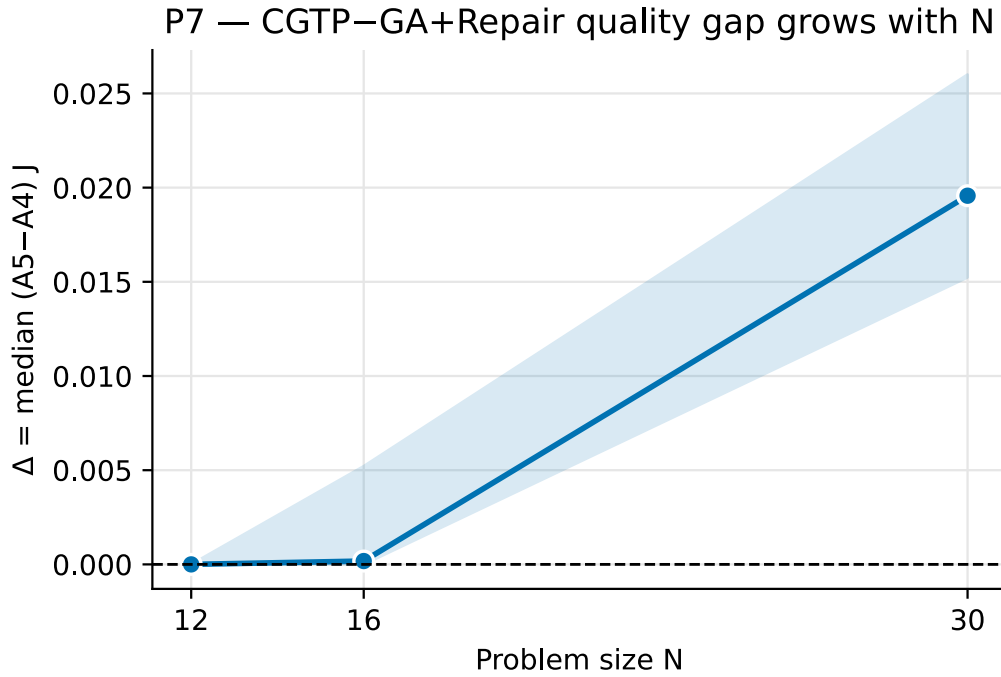


Figure 6: Scaling of the A5–A4 static-quality gap with problem size N : an exact tie at $N = 12$, negligible at $N = 16$, and a small but significant repair-based-GA advantage at $N = 30$.

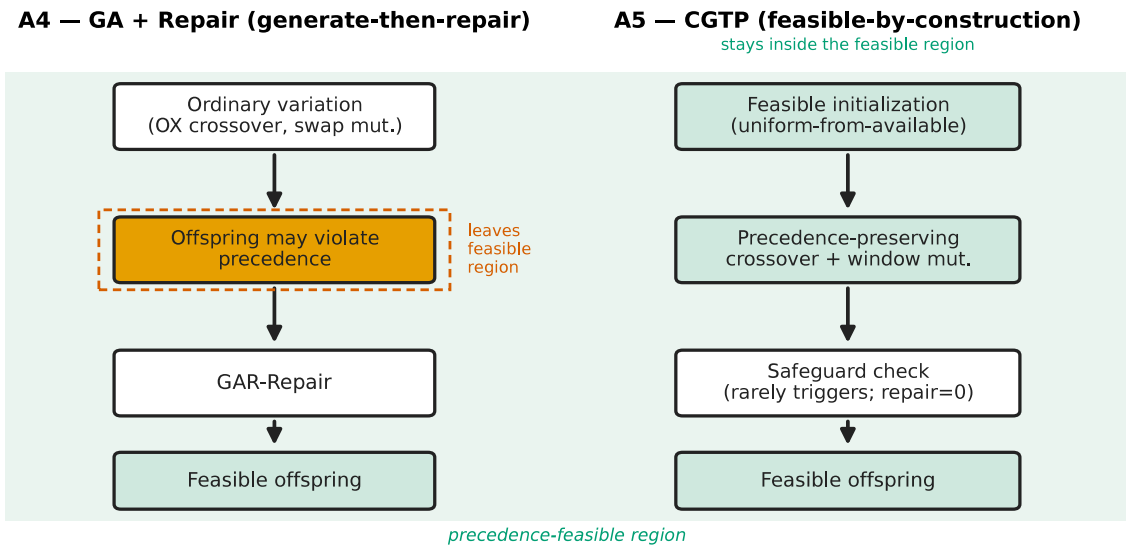


Figure 7: Operator-mechanism schematic — GA+Repair (A4, generate-then-repair) vs CGTP (A5, feasible-by-construction). A4 may leave the precedence-feasible region and repair back into it; A5 stays inside it by construction, so precedence repair is not on the routine search path.

Table 7: Preliminary simulated reachability audit (E5-lite) on N8-CANONICAL with a Franka Emika Panda. Poses are IK-solved with a top-down tool orientation; “reachable” requires joint-limit satisfaction and forward-kinematics error < 8 mm; “collision-free” checks self, board, floor and (for D1) the keep-out box. Preliminary audit only; not a robot-level feasibility claim.

Scenario	Poses audited	Reachable	Collision-free
Static optimum σ^*	8	8/8	8/8
D1 workspace obstruction (remaining) ^a	4	4/4	4/4
D5 recoverable-task-failure suffix ^b	5	5/5	5/5
D2 displaced positions (all admissible) ^c	79	79/79	79/79
Total audited poses	96	96/96	96/96
D1/D3/D4/D5 replanned suffixes	position-preserving: covered by the 8-pose subset argument		

^aThe obstructed cell is deferred during the exclusion window, so its pose is not attempted (four remaining). ^bThe failed plan-head task is only transiently excluded and is still executed after the window, so it stays in the audited suffix (five). ^cThe 79 distinct admissible displaced positions are a superset of the 83 recoverable D2 scenarios’ poses.

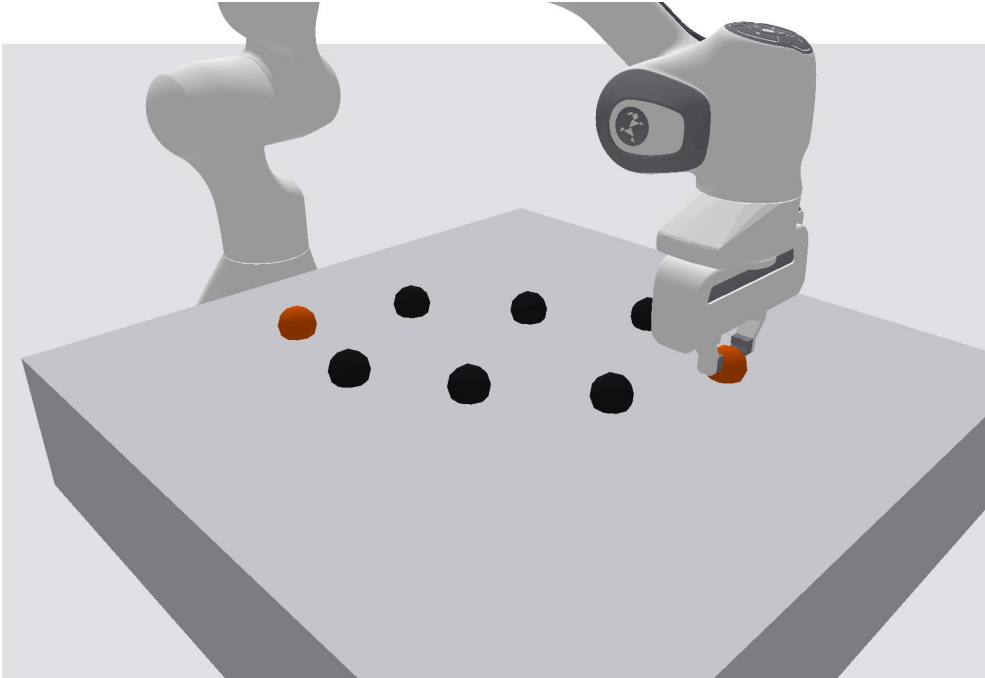


Figure 8: Simulated workcell for the E5-lite reachability audit: a Franka Emika Panda over the eight canonical target cells (source Base and sink G highlighted), tool in a top-down approach. All σ^* poses and all audited replanned-suffix poses were IK-reachable within joint limits and collision-free (Table 7).

5 Discussion

Feasibility preservation and solution-quality dominance are not the same objective, and the results should be read in that light. The clearest task-level benefit of CGTP is that its evolutionary search remains inside the precedence-feasible region by construction, recording zero precedence-repair events; the clearest limitation is that this structural discipline does not translate into a static-cost advantage, and at $N = 30$ repair-based GA is measurably better.

5.1 Static quality: interpretation

Random topological generation and GA+Repair both produce legal sequences reliably, so a legal final order is not what distinguishes the methods. On objective quality, repair-based GA is at least as good as CGTP and better at the largest size. One plausible explanation is that repair-based variation preserves access to a broader pre-repair search neighbourhood, whereas CGTP restricts variation to feasible regions by construction; the broader neighbourhood may aid static objective quality as the problem grows. The present experiments establish the performance pattern but do not isolate this mechanism causally—an operator ablation is required (§5.4). The honest reading is that CGTP’s feasibility preservation is a deliberate trade: it buys a fully precedence-valid search stream and clean partial replanning at a small, size-dependent static-objective premium, not a cost advantage.

5.2 Why recovery is the more important contribution

The recovery results are where the framework is strong. The exact-verification result establishes correctness where it can be checked (128/128 exact suffixes; 3/3 structural cases). At corpus scale, recovery is high and, crucially, **transparently attributed**: with zero budget-conditional search failures, non-recovery is confined to genuinely unrecoverable structural cases certified by an independent oracle rather than absorbed into an undifferentiated "failure" rate. The near-even same/resequenced split shows that adaptation is usually genuine re-sequencing under changed conditions, not compulsory reordering nor trivial continuation—consistent with an execution-state-aware view in which completed work is frozen and only the remaining graph is reconsidered.

5.3 Task-level versus robot-level feasibility

A precedence-valid order is a task-level object, but it is the object a robot execution layer consumes: in an assembly pipeline the planner’s job is to hand the manipulator a legal, disturbance-adapted order to attempt. In that sense the evolutionary task-level decisions studied here directly serve downstream robot execution—the constraint-preserving operators guarantee that every candidate the robot could be given is precedence-consistent, and the execution-state-aware replanner guarantees that, after a disturbance strikes mid-execution, the remaining sequence returned to the robot is still legal with respect to the already-completed prefix. Whether the manipulator can then physically realize every transition is a separate, robot-level question that this study does not answer with formal evidence. Retaining the conceptual distinction—logical sequence validity is necessary but not sufficient for physical executability—motivates the closed-loop extension in §5.4, but no robot-level feasibility result is claimed here.

5.4 Limitations and future work

Limitations. (i) The corpus provides ten instances per cell, which bounds cell-level statistical power (effect sizes and CIs are reported alongside every test). (ii) Static comparisons use a single frozen budget ($B = 10,000$); budget sensitivity is not characterized. (iii) A single injection point ($k_{inj} = \lfloor 0.375N \rfloor$) is used. (iv) Corpus-scale recovery evaluates CGTP only, so recovery is reported as capability and

attribution, not as superiority over another replanner. (v) The disturbance model is a task-level abstraction. (vi) Large-scale global optimality is not established, as exhaustive enumeration is infeasible beyond the canonical case. (vii) **Full motion-level executability** (approach trajectories, sequential transitions and closed-loop feedback) **is not evaluated**; the audit of §4.5 covers pose-level reachability and static collision only, and the planned sequences remain task-level plans.

Future work. (a) An operator ablation to explain the $N = 30$ static gap by separating the effect of feasible-by-construction variation from that of the search neighbourhood. (b) Closing the loop to robot-level feasibility—beyond the preliminary simulated reachability audit of §4.5, feeding motion-level infeasibility (inverse-kinematics, collision and sequential-executability failures) back into task replanning (the full E5 post-validation audit and the E6/E6b execution-time-feedback experiments), which are ongoing extensions on high-performance computing and are not part of the present evidence. (c) Multiple injection points and budget-sensitivity studies. (d) Richer assembly graphs and execution constraints.

6 Conclusion

This paper presented Constraint-Aware Genetic Task Planning (CGTP), a constraint-preserving evolutionary planner with execution-state-aware partial replanning, evaluated under a preregistered, hash-sealed protocol on a canonical benchmark and a nine-cell corpus. CGTP preserves task-level precedence feasibility by construction, recording zero precedence-repair events across the campaign. On static objective quality it is competitive with repair-based GA at $N = 12$ and $N = 16$ but **significantly behind repair-based GA at $N = 30$** ; among the strong search methods, static quality is closely clustered and CGTP is not the best static optimizer. Its contribution is instead reliable constraint-aware adaptation: canonical replanning is **exact in all 128/128 structurally recoverable cases** (with all 3/3 structurally unrecoverable cases correctly flagged, from 131 admissible scenarios), and corpus-scale recovery is high ($D1 \approx 0.978$, $D2\text{--}D4 = 1.000$, $D5 \approx 0.881$) with **no budget-conditional search failures** and structural non-recoverability separated cleanly from search failure. A preliminary simulated reachability audit on the canonical benchmark (96/96 audited poses IK-reachable and collision-free in a Franka Emika Panda workcell) positions the task-level plans within a representative simulated environment. CGTP thus trades a small, size-dependent static-objective premium for structural feasibility preservation and highly reliable disturbance replanning. Full robot-level execution feasibility, and the closed-loop integration of motion feedback into task replanning, are subsequent extensions.

Declarations

Funding. The authors received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Competing interests. The authors declare no competing interests.

Ethics approval. Not applicable; this study did not involve human participants or animals.

Consent to participate / for publication. Not applicable.

Data and code availability. All data and code supporting the findings of this study—the benchmark corpus and frozen-protocol artefacts, the per-run experimental records, the analysis outputs, and the analysis and experiment code—are provided in the Supplementary Information accompanying this article. The data are synthetically generated under the frozen protocol and carry no privacy, consent, or third-party licensing restrictions.

Author contributions. Y. Meng: conceptualization, methodology, software, formal analysis, and writing—original draft. Y. Sun: methodology, validation, and writing—review and editing. M. Balina: software, data curation, and writing—review and editing. All authors read and approved the final manuscript.

References

- [1] Luiz S. Homem de Mello and Arthur C. Sanderson. AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, 1990. doi: 10.1109/70.54734.
- [2] Ikuo Sera, Natsuki Yamanobe, Ixchel G. Ramirez-Alpizar, et al. Assembly planning by recognizing a graphical instruction manual. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. arXiv:2106.00424.
- [3] Romeo M. Marian, Lee H. S. Luong, and Kazem Abhary. A genetic algorithm for the optimisation of assembly sequences. *Computers & Industrial Engineering*, 50(4):503–527, 2006. doi: 10.1016/j.cie.2005.07.007.
- [4] Matan Atad, Jianxiang Feng, Ismael Rodríguez, Maximilian Durner, and Rudolph Triebel. Efficient and feasible robotic assembly sequence planning via graph representation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023. arXiv:2303.10135.
- [5] Lin Ma, Jiangtao Gong, Hao Xu, Hao Chen, Hao Zhao, Wenbing Huang, and Guyue Zhou. Planning assembly sequence with graph transformer. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. arXiv:2210.05236.
- [6] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. arXiv:2309.16909.
- [7] Chang Shu, Anton Kim, and Sanghyun Park. Subassembly to full assembly: Effective assembly sequence planning through graph-based reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 14946–14952, Atlanta, GA, USA, 2025. doi: 10.1109/ICRA55743.2025.11127710.
- [8] Ismael Rodríguez, Korbinian Nottensteiner, Daniel Leidner, Maximilian Durner, Freek Stulp, and Alin Albu-Schäffer. Pattern recognition for knowledge transfer in robotic assembly sequence planning. *IEEE Robotics and Automation Letters*, 5(2):3666–3673, 2020. doi: 10.1109/LRA.2020.2979622.
- [9] James J. Kuffner and Steven M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2000. doi: 10.1109/ROBOT.2000.844730.
- [10] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. doi: 10.1177/0278364911406761.
- [11] Pablo Jiménez. Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, 24(2):235–250, 2013. doi: 10.1007/s10845-011-0578-5.
- [12] Carlos A. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11–12):1245–1287, 2002. doi: 10.1016/S0045-7825(01)00323-1.
- [13] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition, 1996. doi: 10.1007/978-3-662-03315-9.
- [14] Sancho Salcedo-Sanz. A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer Science Review*, 3(3):175–192, 2009. doi: 10.1016/j.cosrev.2009.07.001.

- [15] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317, 1994. doi: 10.1109/ROBOT.1994.351061.
- [16] Sven Koenig and Maxim Likhachev. D* lite. In *Eighteenth National Conference on Artificial Intelligence (AAAI)*, pages 476–483, 2002.
- [17] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic A*: An anytime, replanning algorithm. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 262–271, 2005.
- [18] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. doi: 10.1016/j.artint.2014.11.006.
- [19] Djamila Ouelhadj and Sanja Petrovic. A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, 12(4):417–431, 2009. doi: 10.1007/s10951-008-0090-8.
- [20] Lydia E. Kavradi, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439.
- [21] Ismael Rodríguez, Korbinian Nottensteiner, Daniel Leidner, Michael Kaßecker, Freek Stulp, and Alin Albu-Schäffer. Iteratively refined feasibility checks in robotic assembly sequence planning. *IEEE Robotics and Automation Letters*, 4(2):1416–1423, 2019. doi: 10.1109/LRA.2019.2895845.
- [22] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:265–293, 2021. doi: 10.1146/annurev-control-091420-084139.
- [23] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1470–1477, 2011. doi: 10.1109/ICRA.2011.5980391.
- [24] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, 2014. doi: 10.1109/ICRA.2014.6906922.
- [25] Marc Toussaint. Logic-geometric programming: an optimization-based approach to combined task and motion planning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1930–1936, 2015.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [CGTPsupplementarydatacode.zip](#)