LINKEVO: LEARNING WHEN TO RECOMBINE MODULAR PROMPTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Genetic prompt optimization improves modular language-model systems by evolving instruction bodies while preserving their interfaces, tools, and control flow. However, recombination can disrupt dependencies between modules, and the value of inheriting an instruction block changes with its recipient context and evaluation cost. We introduce LinkEvo, a bounded-population genetic optimizer that measures interactions through controlled block substitutions, builds candidate groups from these measurements, and allocates mutation and crossover effort using context-weighted signed gate gains under an explicit adaptation budget. Across six benchmarks and six executor families, LinkEvo achieved positive observed differences in 34 of 36 settings against uniform genetic-action allocation and 33 against GEPA with merging, with median gains of 1.59 and 0.82 percentage points, respectively, and Qwen3.6 retrieval gains over GEPA with merging of 1.20 points on HotpotQA and 1.33 on HoVer. These results identify a practical role for optimization history in learning which instruction combinations to inherit and when their predicted utility warrants further evaluation, supporting gains across several model families while exposing small mixed differences on reasoning tasks.

1 INTRODUCTION

Modular language-model programs coordinate retrieval, reasoning, code, and tool use through editable instructions with fixed interfaces (Khattab et al., 2024; Opsahl-Ong et al., 2024). Genetic prompt optimizers search instruction populations through mutation and recombination (Guo et al., 2024; Fernando et al., 2024; Agrawal et al., 2026). Their central challenge is preserving useful dependencies between modules while allocating expensive evaluations.

However, useful instruction bodies need not remain useful when moved independently or into a different surrounding program. A summary that preserves an intermediate entity may help only when the next query instruction uses it to retrieve the missing evidence. Inheriting both instructions can

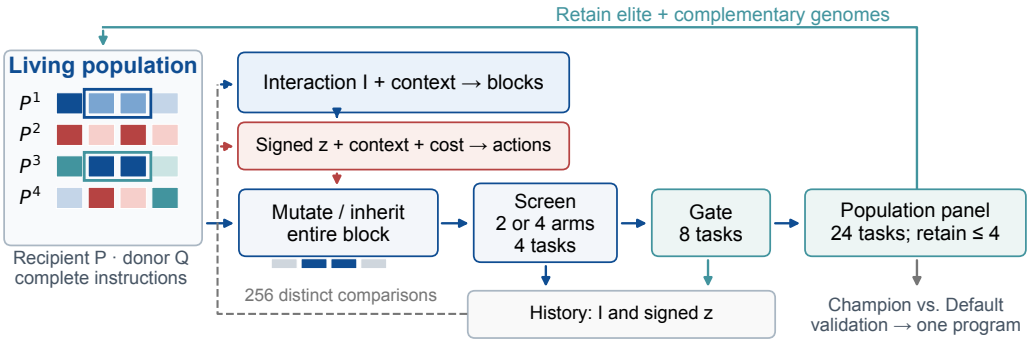


Figure 1: **Learning what to inherit and where to evaluate it.** Up to four living genomes supply recipients and donors. Interaction history defines blocks; contextual signed gains prioritize mutation or crossover. Screening, gating, and population evaluation precede bounded retention. Final selection returns one program; reporting remains outside adaptation.

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.

preserve that dependency; moving either alone can break it. Evaluating every possible combination is expensive, and a successful past edit can become less informative as its recipient changes. GEPA's analysis reports executor-dependent merge effects and attributes the differences to budget allocation and merge timing (Agrawal et al., 2026). The open question is which instruction blocks are worth inheriting, in which recipient contexts, and at what evaluation cost.

We introduce **LINKEVO**, a bounded-population genetic optimizer for fixed modular language-model programs. It uses measured edit interactions to construct candidate instruction blocks and context-weighted signed update gains to allocate work between mutation and crossover (Figure 1). A population of at most four genomes supplies literal donor alleles. Controlled substitutions compare a recipient with partial and complete block replacements; a separate gate measures the utility of the screen-selected child. Interaction records form a linkage hierarchy, while signed gate outcomes rank prospective actions using their context and predicted cost. Current evaluation determines entry into population maintenance.

The core insight is to separate dependence, utility, and relevance. Interaction measures whether two substitutions behave additively; signed update gain measures whether the selected change helps its recipient; contextual similarity determines how strongly an old observation should influence a new action. A strongly interacting block can still be harmful, and a previously useful block can fail in a new background. This separation lets LinkEvo learn both complementary and antagonistic dependencies without treating interaction strength as reward improvement. Population diversity preserves alternative instruction combinations during adaptation, and final selection returns one genome with the original interfaces and control flow.

We evaluate six benchmarks spanning retrieval, reasoning, code, and tool use with six executor families and a fixed proposer. In the Qwen3.6 retrieval programs, LinkEvo exceeds uniform genetic-action allocation by 2.40 points on HotpotQA and 3.67 on HoVer, and GEPA with merging by 1.20 and 1.33 points. Across all 36 task–executor cells, the respective positive-difference counts are 34 and 33. Gains are smaller on proprietary executors, and the reasoning benchmark has near-zero mixed differences. Component controls separate grouping, contextual utility, and crossover from population search, while resource accounting charges screening, gating, population maintenance, proposals, and final selection.

Our contributions are:

- **Measured linkage for genetic variation.** Controlled module substitutions propose blocks that can be mutated or inherited together.
- **Contextual allocation with a distinct utility target.** Signed gate gains prioritize genetic actions under an explicit cost model; interaction statistics supply grouping information.
- **Evidence defining the domain of advantage.** Matched controls, established optimizers, task–executor comparisons, and full adaptation accounting identify where measured dependencies improve search.

2 RELATED WORK

Genetic and modular prompt optimization. DSPy represents language-model systems through modules and signatures; MIPRO optimizes instructions and demonstrations within multi-stage programs (Khattab et al., 2024; Opsahl-Ong et al., 2024). Population search, mutation, and crossover are established prompt-optimization tools. EvoPrompt combines language-model proposal generation with evolutionary operators, and Promptbreeder evolves both task prompts and the mutation instructions used to revise them (Guo et al., 2024; Fernando et al., 2024). GEPA couples reflective mutation with diverse candidate selection and offers a system-aware merge variant (Agrawal et al., 2026). Its merge analysis reports different effects across executors and motivates studying how to allocate mutation and crossover. Our instruction-only comparisons identify the adaptations to MIPROv2 and GEPA in Appendix E.

Linkage learning and block inheritance. Gene-pool optimal mixing uses learned variable dependencies to guide joint inheritance, and conditional GOMEA further restricts mating according to dependencies (Dushatskiy et al., 2024). This establishes the broader rationale for preserving coordinated components during recombination. LinkEvo measures outcome interactions through controlled

substitutions of module instructions, then builds a recipient-specific hierarchy from those observations. Its living population supplies complete instruction alleles, while the surrounding program determines the context in which a block is tested. The history summarizes observed substitutions rather than all possible interactions among instruction texts.

Contextual allocation under changing evidence. COBRA-Skills prioritizes evolving textual candidates using execution feedback, a learned reward predictor, and a contextual-bandit exploration term (Lu et al., 2026). Non-stationary bandit methods use sliding windows to limit reliance on older observations as conditions change (Cheung et al., 2019). LinkEvo uses a transparent age-and-context kernel over signed gate gains to rank genetic actions and separately uses interaction measurements for grouping. Its support bonus and cost predictor are scheduling heuristics. The contribution is this division of information from controlled block substitutions between inheritance and allocation.

Adaptive selection and independent reporting. Model-selection criteria can overfit repeated evaluations (Cawley & Talbot, 2010). LinkEvo's population panel and gates are adaptive training components; final selection compares two frozen candidates on validation, followed by reporting on a separate task set. Paired run-level intervals describe uncertainty conditional on the reporting tasks, and cross-benchmark summaries retain each task's metric and weighting.

3 METHOD

3.1 PROGRAM REPRESENTATION

A genome $P = (p_1, \ldots, p_m)$ contains the complete instruction body, or allele, at each editable locus. A locus is an instruction slot and can be invoked repeatedly. Executor weights and settings, module signatures, tools, retrieval configuration, and control flow remain fixed within a task–executor cell. An execution returns binary reward $R(P, x, u)$ for task x and recorded realization u. The objective is the held-out reward of the final selected genome under an adaptation budget. Reporting outcomes are unavailable to search.

We distinguish immutable Default P_0, living population $\mathcal{P}_t$, recipient P, donor Q, and population-panel champion P_{champ}. A realization key identifies an execution record; it does not imply reproducible random draws across API requests or service revisions. Exact cache reuse preserves the identity of the original observation.

3.2 POPULATION AND VARIATION

The population starts at $\{P_0\}$ and contains at most four distinct genomes. Each member has a reward vector on the same fixed 24-task training panel. Retention first keeps the highest-mean elite, breaking ties by shorter total instruction length and then earlier creation index. Up to three further members are selected greedily by the number of panel tasks they solve that the retained set leaves unsolved. Ties favor larger minimum module-level Hamming distance to retained genomes, higher mean reward, shorter instructions, then earlier creation. The same tie rules fill slots when coverage gains are zero. Reduction considers only living members and the newly scored child, deduplicates genomes, and recomputes the elite. Every child-panel evaluation is charged even if retention discards the child.

Recipient selection gives half the probability to the elite and divides the other half uniformly among non-elites; a sole member is selected deterministically. We condition this distribution on recipients with a resource-feasible action. Retired genomes remain in the lineage archive and cannot donate.

An action $a = (P, o, S, \pi, Q)$ specifies mutation or crossover o, a block S with $1 \leq |S| \leq \min(4, m)$, and a two-part partition π for multi-locus blocks. Mutation uses one charged proposal call to replace every body in S. Crossover copies those bodies literally from a single living donor Q that differs from P at *every* selected locus. The remaining bodies stay fixed. Identical crossover payloads for the same recipient, block, and partition form one action, represented by the earliest-created equivalent donor. Mutation responses must contain exactly the requested keys and complete, changed, nonempty bodies of at most 1,024 local tokens each. Invalid responses consume their call and attempt. Appendix C.2 gives the literal instruction and evidence policy.

3.3 CONTROLLED COMPARISONS

For $|S| \geq 2$, let $\pi = (A, B)$ contain disjoint nonempty parts with union S. The four arms are recipient P_{00}, substitutions at A only (P_{10}), at B only (P_{01}), and at both (P_{11}). They share four screen tasks. Singletons compare only parent and child. For four-arm mean rewards y_{ab}, define

$$I = y_{11} - y_{10} - y_{01} + y_{00}, \qquad J = y_{11} - \max\{y_{00}, y_{10}, y_{01}\}, \qquad (1)$$
$$G = \max\{y_{00}, y_{10}, y_{01}, y_{11}\} - y_{00}.$$

Interaction I supplies linkage information; joint advantage J and available gain G are diagnostics. The additive reference can lie outside $[0, 1]$, and adaptive selection affects the interpretation of all three sample statistics.

The highest-reward screen arm wins; ties prefer fewer changed loci and then canonical order $00, 10, 01, 11$. A parent win has no-update utility $z_r = 0$ and skips the gate. Otherwise, freeze the winner and evaluate it against the parent on eight other training tasks:

$$z_r = \frac{1}{8} \sum_{x \in T_{g,r}} \left[R(P_{w,r}, x, u_{w,x}) - R(P_{00,r}, x, u_{0,x}) \right]. \qquad (2)$$

A strictly positive gate gain admits the child for population-panel evaluation and bounded retention. Negative gains remain in history. This adaptive gate measures local update utility; independent reporting evaluates the final program. Singleton records never update linkage edges.

A dependency worth inheriting. In the two-locus retrieval case in Appendix G, a donor summary preserves intermediate entities and its query instruction targets the missing relation. On 100 reporting tasks, the recipient, summary-only, query-only, and joint substitutions solve 50, 47, 46, and 64 tasks. The joint change has $I = 0.21$ and $J = 0.14$: transferring both bodies preserves a useful dependency that either partial substitution disrupts.

3.4 CONTEXTUAL HISTORY AND LINKAGE

The numerical archive holds the 256 most recent distinct completed comparisons. Each stores environment, operator, recipient/donor and replacement identities, block and partition, screen/gate request keys and outcomes, winner, signed utility, disposition, and usage provenance. Exact replays reuse the original record without refreshing its age or adding weight. Different records may share outcomes.

For a candidate action, compatible evidence receives weight

$$w_r(a) = \mathbf{1}\{e_r = e_a, o_r = o_a, S_r = S_a, \pi_r = \pi_a\} \exp\left[-\frac{t - t_r}{16} - 2d_{\mathrm{bg}} - 2d_{\mathrm{loc}} \right]. \qquad (3)$$

Here t counts completed comparisons and t_r is the original completion index. Environment e fixes benchmark/scorer, executor, contracts, and execution configuration. The background distance is the fraction of loci outside S with different body hashes, or zero for an empty complement. Body distance is one minus Jaccard similarity of whitespace word-five-gram sets; shorter bodies contribute their whole sequence as one element. Local distance averages recipient-body distances over S for mutation and both recipient- and donor-body distances for crossover. Mutation scores exclude the ungenerated proposal. Singleton partitions have one canonical null value. History stays within its task–executor environment.

Linkage uses recipient context independently of a candidate donor:

$$\widetilde{w}_r(P; i, j) = \mathbf{1}\{e_r = e_P, S_r = \{i, j\}, \pi_r = \{\{i\}, \{j\}\}\} \exp\left[-\frac{t - t_r}{16} - 2d_{\mathrm{bg}} - 2d_{\mathrm{rec}} \right],$$
$$\bar{I}_{ij}(P) = \frac{\sum_r \widetilde{w}_r(P; i, j) I_r}{1 + \sum_r \widetilde{w}_r(P; i, j)}, \qquad L_{ij}(P) = |\bar{I}_{ij}(P)|. \qquad (4)$$

Here d_{bg} compares loci outside $\{i, j\}$ and d_{rec} averages recipient-body distances at i, j; pair records pool both operators. Complementary and antagonistic dependencies both contribute grouping strength,

although opposite signs can cancel. Larger-block measurements never create individual-pair observations.

Starting from singleton clusters, repeatedly merge the pair with maximum mean cross-edge L_{ij}, with lexicographic locus-tuple ties. Exploitation uses hierarchy nodes of size at most four, all singletons, and all pairs. A hierarchy node inherits its child partition; a pair uses its singleton partition. Other exploration blocks split their sorted locus IDs after $\lfloor |S|/2 \rfloor$. Canonical ordering makes the partition part of action identity. Unobserved edges have zero prior strength and an unobserved flag.

3.5 Allocation and accounting

Context-weighted signed gains and support mass define

$$\widehat{\mu}_a = \frac{\sum_r w_r(a) z_r}{1 + \sum_r w_r(a)}, \quad s_a = \sum_r w_r(a), \quad U(a) = \frac{\widehat{\mu}_a + 0.10/\sqrt{1 + s_a}}{\max(1, \widehat{c}_a/1000)}. \tag{5}$$

Rewards use fractions. The bonus is a heuristic based on weighted support, and $\widehat{c}_a$ predicts incremental processed tokens. With probability 0.20, exploration samples an available operator, eligible size, eligible block, and crossover donor uniformly at each stage, conditioned on feasibility. It can reach any block of size at most four. Otherwise, maximize U over exploitation actions, breaking ties by lower predicted cost and then the recorded random stream.

The cost predictor uses an exponential moving average of actual noncached executor tokens per episode, updated at rate $1/8$ and initialized by eight ordinary episodes. Multiply by the worst-case action allowance of 48 singleton or 56 block episodes, then add the proposal-token moving average for mutation. Its initial value is 14,336 tokens; charged invalid attempts with known usage also update it. Literal crossover adds no proposal tokens. The predictor assumes no future cache hits and guides scheduling rather than reserving tokens. Missing usage makes this allocation configuration incomplete.

Each run permits 2,232 physical episodes, 132 attempted proposal calls, and 768 action attempts. Initialization uses eight ordinary episodes and 24 Default population-panel episodes; reserving 600 for final selection leaves 1,600 for search. Before an action, reserve 48 or 56 episodes for screen, possible gate, and possible child-panel evaluation. Only cache misses debit episodes, and unused reservation is released. Numerical allocation makes no controller-model call. Appendix C.4 separates analytical reservations from measured usage.

3.6 Complete algorithm

Algorithm 1 combines these operations. All attempts, including invalid proposals and cached replays, count toward termination. Infrastructure failures preserve partial usage and return incomplete status; malformed task outputs receive the declared failure reward. After search, only the training-panel champion and immutable Default enter final selection on 300 validation tasks. Default wins ties. Exact reuse needs 300 physical episodes when they are identical; otherwise the reservation covers both. Reporting evaluates the frozen selected genome afterward. Deployment uses one program with the original interfaces.

4 Results

LinkEvo reaches **64.40% HotpotQA exact match** and **56.27% HoVer document success** on the Qwen3.6 retrieval programs, with gains over uniform genetic allocation of **2.40** and **3.67 percentage points**.

4.1 Evaluation setting

We evaluate six benchmarks spanning retrieval, reasoning, code, and tool use with six executor families: Qwen3.6-35B-A3B, Gemma-4-31B-it, DeepSeek-V4.1-Flash, GPT-5.5, Claude Sonnet 5, and Gemini 3.6 Flash. The primary proposer remains Qwen3.6-35B-A3B across executors. Comparisons are within fixed task–executor cells, with the same instruction-only artifact and compatible proposer access.

Algorithm 1 LinkEvo: contextual genetic optimization

Require: Default P_0, fixed environment and task splits; episode/proposal/attempt limits $(2232, 132, 768)$

Initialize exact cache, lineage archive, distinct-comparison archive
Charge 8 ordinary episodes and 24 population-panel evaluations of P_0
$\mathcal{P} \leftarrow \{P_0\}$; reserve 600 episodes for final selection; $k, t \leftarrow 0$
while $k < 768$ **do**
    Identify recipients with a resource-feasible action; stop if none remain
    Select recipient by the conditioned elite/non-elite distribution
    Build linkage hierarchy and eligible action descriptors
    Select action by exploration or U; reserve 48/56 episodes; $k \leftarrow k + 1$
    if mutation **then**
        Charge a proposal attempt; request and parse all selected alleles
        if invalid **then**
            Record usage; release reservation; **continue**
        end if
    else
        Copy selected donor alleles exactly
    end if
    Screen 2 singleton or 4 block arms; freeze the winner by the tie rule
    if parent wins **then**
        $z \leftarrow 0$; gate status $\leftarrow$ not reached
    else
        Gate winner against parent on 8 screen-disjoint tasks; record signed z
        if $z > 0$ **then**
            Evaluate winner on the 24-task population panel
            Add genome and lineage; deduplicate and retain at most 4 genomes
        end if
    end if
    $t \leftarrow t + 1$; retain distinct comparison once; update usage and cost predictors
    Release unused reservation
end while
Select population-panel champion P_{champ}
Compare P_{champ} with P_0 on validation; choose P_0 on a tie
Freeze selected genome and evaluate on held-out reporting tasks

HotpotQA and HoVer measure answer exact match and three-document supporting-title retrieval, respectively (Yang et al., 2018; Jiang et al., 2020). The broader suite adds MuSiQue answerable-subset EM, BBEH arithmetic category-macro accuracy, LiveCodeBench code-generation pass@1, and BFCL multi-turn task success (Trivedi et al., 2022; Kazemi et al., 2025; Jain et al., 2025; Mao et al., 2024). Appendix A.3 distinguishes these metric scopes. Every cell reserves 300 validation tasks for final selection. The two primary retrieval programs use 150 training and 300 reporting tasks over five runs. The 2,232-episode adaptation cap includes initialization, search, population maintenance, and final selection; measured calls and tokens provide separate resource views.

4.2 Main results

Table 1 compares LinkEvo with matched genetic controls and instruction-only configurations of MIPROv2, GEPA, and GEPA with merging. LinkEvo exceeds utility-neutralized allocation by 2.40/3.47 points on HotpotQA/HoVer and GEPA+Merge by **1.20/1.33 points**. Gains over Default are 9.73/12.27 points. The complete-optimizer comparisons establish the final system's performance, while matched controls identify which parts of the genetic search contribute. Against GEPA+Merge, the conditional paired 95% run-level intervals are [0.65, 1.76] points on HotpotQA and [0.51, 2.16] on HoVer. Appendix D gives the counts, derivation, and uncertainty scope.

4.3 Resource use

We compare reward against the full cost of producing a selected program. Table 2 shows that LinkEvo uses slightly more physical episodes and fewer processed tokens than Uniform and GEPA+Merge at the common episode cap. For HotpotQA, its 29.10 million tokens compare with 29.58 and 30.36 million; HoVer uses 30.24 million versus 30.87 and 31.92 million. Population evaluation and invalid

Table 1: Reporting reward for the Qwen3.6-35B-A3B retrieval programs (%). Each cell averages five runs on 300 reporting tasks under a 2,232-episode adaptation cap. Every method pays for its acquisition, proposal, population-maintenance, and final-selection work. Bold marks the largest observed mean. Controls and complete-optimizer adaptations are defined in Appendices A and E.

Method	HotpotQA EM	HoVer retrieval
Default	54.67	44.00
Singleton-only genetic search	61.46	51.92
Uniform genetic allocation	62.00	52.60
Utility-neutralized allocation	62.00	52.79
Random linkage hierarchy	62.60	53.60
Context-free history	63.26	54.66
Mutation only	63.48	55.13
Full11 genetic screening	62.72	54.13
Full16 genetic screening	63.14	54.68
MIPROv2 (instruction-only)	63.06	54.59
GEPA (instruction-only)	63.13	54.66
GEPA+Merge (instruction-only)	63.21	54.94
LINKEVO	**64.40**	**56.27**

proposal attempts are included. These quantities compare actual within-cell usage and preserve the distinction between episodes, model calls, and tokens.

Table 2: Measured adaptation resources, averaged over five runs. Episodes include initialization, screening, gating, population evaluation, and final selection; proposal attempts include invalid responses. These retrieval episodes complete four executor calls. Tokens sum actual input and output usage across roles.

Task	Method	Episodes	Proposals	Model calls	Tokens (M)
HotpotQA	LINKEVO	2,204.0	18.4	8,834.4	29.10
HotpotQA	Uniform	2,195.2	20.6	8,801.4	29.58
HotpotQA	GEPA+Merge	2,190.4	27.2	8,788.8	30.36
HoVer	LINKEVO	2,207.2	19.6	8,848.4	30.24
HoVer	Uniform	2,189.6	21.4	8,779.8	30.87
HoVer	GEPA+Merge	2,192.0	28.0	8,796.0	31.92

Figure 2 shows how the reward gap develops across five caps. At 816 episodes, LinkEvo and GEPA+Merge are nearly matched; at 2,232, their gap grows to 1.20 points on HotpotQA and 1.33 on HoVer. LinkEvo's advantage over Uniform grows more strongly, reaching 2.40/3.67 points. The curves connect separately validated budgets and show the value of allocating additional search effort as history and populations develop.

4.4 SEARCH BEHAVIOR

We examine the genetic actions and the outcomes that consume the budget. Figure 3 separates completed mutation/crossover comparisons from invalid proposal attempts. HotpotQA completes 300 comparisons, including 84 mutations and 216 crossovers; HoVer completes 287, including 91 mutations and 196 crossovers. The additional 8/7 invalid proposals are charged attempts. The gate admits 70 HotpotQA and 67 HoVer children for population evaluation. Their changed-locus counts are 28/28/10/4 and 24/29/10/4 for sizes one through four (Appendix B.1). Singleton children remain useful, while three- and four-locus changes exercise inheritance beyond pairs. Operator frequency describes allocation; the mutation-only control measures the effect of removing crossover. Retention is a later decision, so gate-passing counts are not population-survival counts.

4.5 MECHANISM EVIDENCE

We separate grouping, contextual utility, and recombination through the controls in Table 1. Neutralizing signed utility reduces scores to 62.00%/52.80%; random hierarchy yields 62.60%/53.60%;

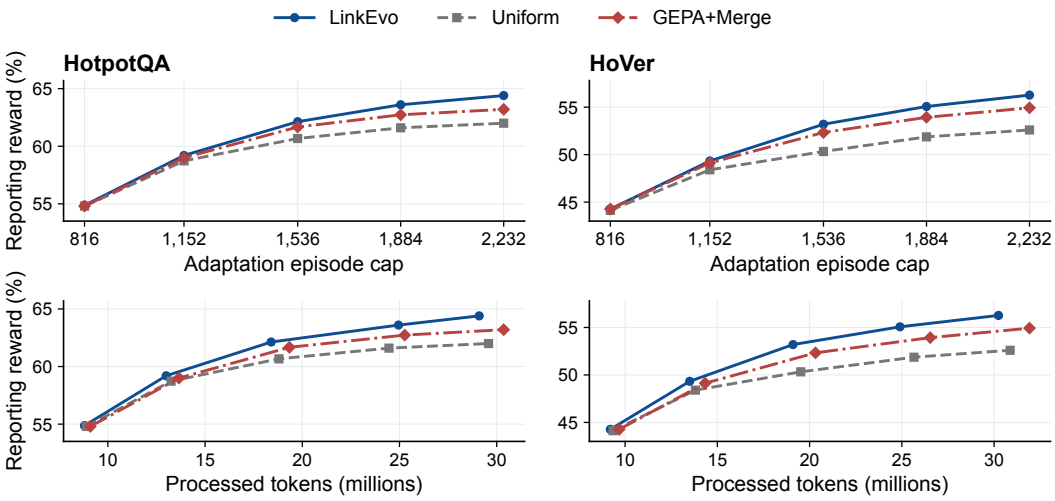


Figure 2: **Reward as a function of adaptation budget.** Each point averages five separately capped runs. Upper axes use assigned episode caps; lower axes use measured processed tokens. Population maintenance and final selection are included. Final coordinates match Tables 1 and 2.

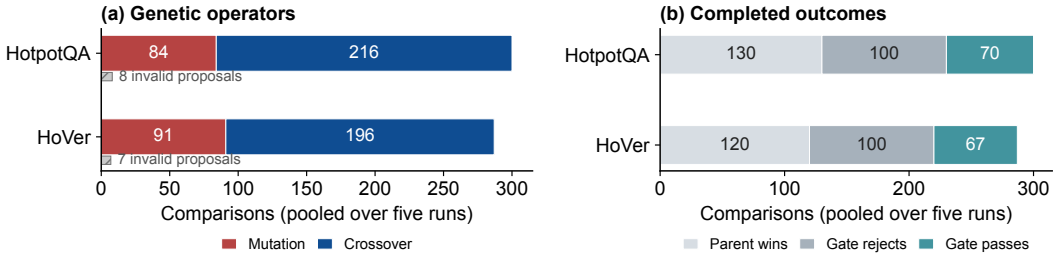


Figure 3: **Genetic search behavior over five runs.** Completed operator counts and invalid proposal attempts are separate. Completed comparisons partition into parent screen wins, gate rejections, and gate-passing children. Passing children undergo population-panel evaluation and need not survive bounded retention.

context-free history gives 63.26%/54.67%; mutation-only search reaches 63.47%/55.13%. Every intervention lowers the observed mean on both retrieval tasks. Their effects are not additive because each changes later populations and histories.

Local diagnostics in Table 8 fix the recipient, operator, and replacement material. LinkEvo gains 3.67/4.08 points, compared with 1.92/2.00 for uniform local group selection. Full16 finds larger local gains, 5.50/5.67 points, by evaluating a broader pool. Its lower online scores reflect the importance of charging that additional screening. Shared-checkpoint access controls separately measure inherited records and cache reuse (Table 9). The prompt case in Appendix G shows the mechanism directly: inheriting a coordinated summary/query block succeeds where either partial substitution reduces reward.

4.6 Transfer and Boundaries

We repeat optimization within each task–executor cell to test how the advantage generalizes across program and model families. Figure 4 contains all 36 paired differences against each comparator. LinkEvo is positive in **34/36 cells against Uniform** and **33/36 against GEPA+Merge**; descriptive medians are **1.59** and **0.82 points**, and means are 1.76 and 0.83 points. Retrieval, code, and tool-use programs show positive differences throughout the grid. Proprietary-executor gains are generally smaller, and BBEH has near-zero mixed differences: GPT is negative against both comparators, Claude ties Uniform and trails GEPA+Merge, and DeepSeek ties GEPA+Merge. The pattern identifies

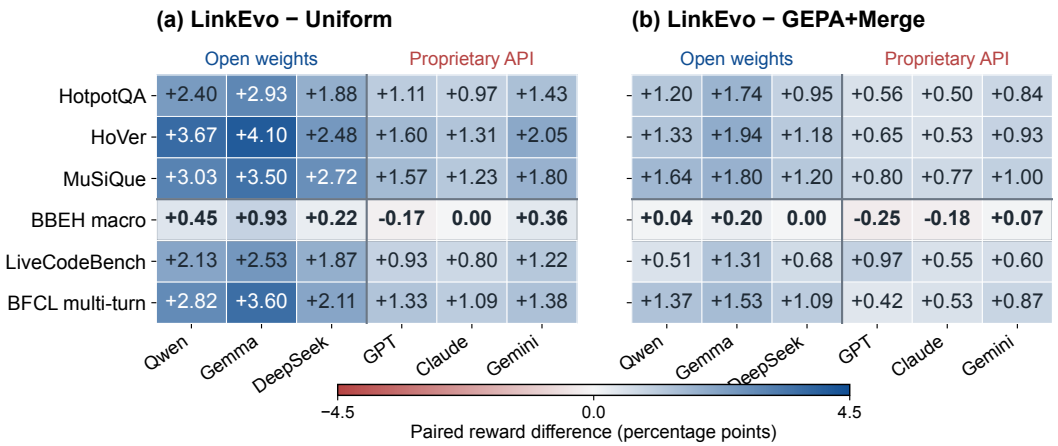


Figure 4: **Task and executor boundaries of the advantage.** LinkEvo minus Uniform (left) and GEPA+Merge (right), in percentage points, on a shared zero-centered scale. Open-weight and proprietary executors are grouped; the horizontal separator separates retrieval from reasoning, code, and tool use. Negative and tied BBEH cells remain visible.

useful scope without collapsing distinct metrics into an absolute score. Each benchmark receives equal weight in the summaries, giving retrieval greater representation than each other task family. Appendix F lists every contrast alongside the primary retrieval anchors.

5 Limitations

Historical state and population evaluation add costs that may outweigh their benefit when loci interact weakly, dependencies change rapidly, or optimization headroom is small. The lexical context kernel approximates relevance, pair-derived hierarchies can miss purely higher-order effects, and a four-member population can discard useful diversity. Reused training panels introduce adaptive selection effects. A four-arm block screen limits current alternatives; full search cost still depends on module count and the actions explored.

The evaluation supports the declared task, program, executor, proposer, and budget configurations. BBEH has small mixed differences, and the cross-cell summaries give retrieval more representation than each other task family. Five-run intervals condition on fixed reporting tasks. Proprietary service changes can limit reproducibility.

6 Conclusion

LinkEvo connects measured edit dependencies with genetic variation. A bounded population supplies instruction blocks for inheritance, interaction measurements define candidate groups, and contextual signed utility allocates mutation and crossover effort. The comparisons show useful gains over uniform genetic allocation and established optimizers across several program families, with small mixed reasoning differences delimiting the advantage. History is useful when its guidance repays acquisition and population-maintenance costs.

AI USE STATEMENT

In this work, we used generative AI tools for implementing the methods. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for summarising or analysing existing literature, sourcing/searching for information, identifying relevant literature, creating or modifying scientific figures or images, editing a research paper to improve readability, and suggesting a structure for a research paper. We have reviewed all AI-assisted work. Specifically, the AI-generated code implementation was thoroughly verified and tested for correctness by the authors, the literature search results and paper summaries were manually cross-referenced with the original source texts to ensure factual accuracy, and the AI-generated figures and structural revisions were carefully reviewed to ensure they faithfully and accurately represent our empirical findings. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We have carefully considered the ethical implications of this work in alignment with the ICLR Code of Ethics. All datasets and pre-trained models utilized in our evaluation are publicly accessible and used in strict compliance with their respective licenses. No private data or personally identifiable information (PII) were collected or processed. While our findings aim to optimize model capabilities—-which inherently carries a broader systemic risk of downstream misuse (such as generating biased or misleading content)—-our specific method is designed purely for research benchmarking and does not actively amplify these vulnerabilities

REPRODUCIBILITY STATEMENT

Section 3 defines the optimizer, action eligibility, retention, contextual history, allocation, and budgets. Appendices A–C specify configuration, controls, program contracts, and accounting. Appendix D defines aggregation and uncertainty; Appendix E distinguishes complete-optimizer adaptations. Appendices F–H provide the comparison grid, prompt case, and comparison scopes.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(70):2079–2107, 2010. URL https://jmlr.org/papers/v11/cawley10a.html.

Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to optimize under non-stationarity. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1079–1087, 2019. URL https://proceedings.mlr.press/v89/cheung19b.html.

Arkadiy Dushatskiy, Marco Virgolin, Anton Bouter, Dirk Thierens, and Peter A. N. Bosman. Parameterless gene-pool optimal mixing evolutionary algorithms. *Evolutionary Computation*, 32(4):371–397, 2024. doi: 10.1162/evco_a_00338. URL https://arxiv.org/abs/2109.05259.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544, 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9156b0f6dfa9bbd18c79cc459ef5d61c-Abstract-Conference.html.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.309. URL https://aclanthology.org/2020.findings-emnlp.309/.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. BIG-Bench Extra Hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26473–26501, 2025. doi: 10.18653/v1/2025.acl-long.1285. URL https://aclanthology.org/2025.acl-long.1285/.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f1cf02ce09757f57c3b93c0db83181e0-Abstract-Conference.html.

Pingchen Lu, Xiangyi Wang, Xiang Li, Jie Mao, Zikun Qu, Junfeng Luo, Yao Shu, Bryan Kian Hsiang Low, and Zhongxiang Dai. COBRA-Skills: Contextual bandit-guided evolution for agent skill optimization. arXiv preprint arXiv:2609.11682, 2026. URL https://arxiv.org/abs/2609.11682v1. Version 1, submitted 10 September 2026.

Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Jason Huang, Vishnu Suresh, Yixin Huang, Xiaowen Yu, Joseph E. Gonzalez, and Shishir G. Patil. BFCL V3: Multi-turn & multi-step function calling, 2024. URL https://gorilla.cs.berkeley.edu/blogs/13_bfcl_v3_multi_turn.html.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

A CONFIGURATION AND EVALUATION PROTOCOL

A.1 CANONICAL CONFIGURATION

Table 3: Canonical LinkEvo configuration.

Parameter	Setting
Population cap	4 distinct living genomes
Population panel	24 fixed training tasks
Ordinary initialization	8 distinct training-task executions
Comparison archive	256 most recent distinct completed records
Numerical admission	Record count
Maximum block size	$\min(4, m)$
Screen / gate	4 / 8 tasks, disjoint within each action
Age-decay scale	16 completed comparisons
Background / local coefficients	2 / 2
Exploration probability	0.20
Allocation bonus	0.10 in reward-fraction units
Executor / proposal cost EMA rate	$1/8$
Proposal evidence	At most 8 traces; 6,144 input tokens
Replacement-body limit	1,024 local-tokenizer tokens per locus
Proposal response allowance	8,192 tokens, including provider-counted reasoning
Physical adaptation cap	2,232 episodes
Proposal-call limit	132 attempted calls, including invalid responses
Action-attempt limit	768, including invalid and cache-served attempts
Final-selection reservation	Up to 600 episodes: two programs on 300 tasks
Deployment	One selected genome

These constants define the evaluated configuration. Instruction length uses a fixed local tokenizer for body limits and population tie-breaking. Provider usage retains its own token units; the two accounting conventions serve different purposes.

Population retention and recipient feasibility. The elite maximizes mean reward on the fixed training panel, then minimizes total instruction length and creation index. Each additional slot maximizes newly covered successful tasks, then minimum module-level Hamming distance to retained members, mean reward, negative length, and negative creation index. When new coverage is zero, the same tie order fills the remaining slots. Exact duplicates share one slot. A recipient with no eligible, affordable action has zero conditioned selection probability. The search stops when no recipient remains feasible. This greedy coverage rule maintains bounded diversity rather than computing an exact Pareto frontier.

A.2 EXECUTOR AND PROPOSER DEFINITIONS

Table 4: Executor references. The fixed proposer for the primary comparisons is Qwen3.6-35B-A3B.

Label	Reference	Access
Qwen	Qwen3.6-35B-A3B	Open weights
Gemma	gemma-4-31B-it	Open weights
DeepSeek	DeepSeek-V4.1-Flash	Open weights
GPT	gpt-5.5	API
Claude	claude-sonnet-5	API
Gemini	gemini-3.6-flash	API

Open-weight references belong to the Qwen, google, and deepseek-ai organizations, respectively. Each comparison holds the executor and proposer fixed within a task cell. Compatible baselines receive the same proposer access. These are six executor families; the common proposer does not

add another family. Model and serving settings, program contracts, and scorer identity are part of the environment compatibility condition in Equation (3).

A.3 Tasks, panels, and metrics

Table 5: Task scope and reporting metrics. The BBEH arithmetic category mean and BFCL multi-turn subset are custom evaluation scopes.

Benchmark	Metric	Scope
HotpotQA	Official answer exact match	Custom reporting split; fixed retrieval program
HoVer	All-supporting-title success	Three-document subset; retrieved-title union
MuSiQue	Answer exact match	Answerable subset with a fixed retrieval/corpus setting
BBEH	Arithmetic mean of category accuracies	Category macro accuracy
LiveCodeBench	Code-generation pass@1	Dated problem window and final-code evaluator
BFCL	Multi-turn task success	Specified multi-turn subset

The primary Qwen3.6 HotpotQA and HoVer comparisons use 150 training, 300 validation, and 300 reporting tasks with run labels 11, 23, 47, 71, and 101. The 24-task population panel is fixed within training, leaving 126 tasks for ordinary initialization, screens, and gates. At each action, a permutation of those 126 IDs supplies four screen tasks followed by eight distinct gate tasks. Initialization takes eight tasks from the same pool; later actions may reuse training tasks. Final selection reserves 300 tasks in every task–executor cell. The population panel and disjoint action panels require at least 36 training tasks in each benchmark.

The official HotpotQA scorer lowercases, removes punctuation and articles, and normalizes whitespace; we add no Unicode NFD transformation under that metric name. HoVer requires all supporting titles in the union of retrieved titles, permits extras, and evaluates retrieval rather than the claim-verification label. The title-normalization and document-mapping operation is fixed within each comparison cell. BBEH uses an arithmetic category mean; the benchmark's published adjusted harmonic aggregation is a different metric (Kazemi et al., 2025). Final code is judged under the LiveCodeBench evaluator; BFCL uses the multi-turn success evaluator (Jain et al., 2025; Mao et al., 2024).

Population-panel tasks are optimization data. Validation and reporting are disjoint from all optimization data and from one another. Reporting labels and evaluator feedback never enter mutation, grouping, allocation, gating, or retention. Release, task IDs and order, filters, random-generator identity, overlap rules, and scorer revisions determine the precise evaluation cell.

A.4 Comparator definitions

Table 6: Matched genetic controls. Every row retains the remaining population, gate, accounting, and final-selection rules.

Comparator	Intervention
Default	Return immutable P_0 without adaptation.
Singleton-only genetic search	Restrict eligible blocks to one locus.
Uniform genetic allocation	Set exploration probability to one, using the operator/size/block/-donor sampler over all eligible blocks. The computed hierarchy does not restrict that sampler.
Utility-neutralized allocation	Set $z_r = 0$ only in $\widehat{\mu}_a$; retain support mass, linkage I, context weights, exploration, and cost terms.
Random linkage hierarchy	Randomly permute locus labels on the learned hierarchy each decision, preserving shape and block sizes; retain all singleton and pair actions.

Comparator	Intervention
Context-free history	Set age and distance factors to one in both kernels; preserve environment and operator/block/partition compatibility, archive capacity, and deduplication.
Mutation only	Remove crossover actions and keep the four-genome population.
Full11 genetic screening	For four loci, propose a fresh full mutation batch or use a donor differing at every locus; screen parent, four singletons, and six pairs.
Full16 genetic screening	Use the same full-batch construction and screen all sixteen subsets.

Full11 and Full16 reserve $44 + 16 + 24 = 84$ and $64 + 16 + 24 = 104$ episodes per action. Each mutation action draws a fresh full batch. Their broader generation and screening scope makes them system comparisons. Initialization follows the common 32-episode rule.

Interventions can change later populations and histories. Utility neutralization preserves interaction-based grouping, while random hierarchy changes grouping and partitions. Component effects therefore describe separate online systems and are not additive contributions. Instruction-only adaptations of complete optimizers are detailed in Appendix E.

A.5 Diagnostic state provenance

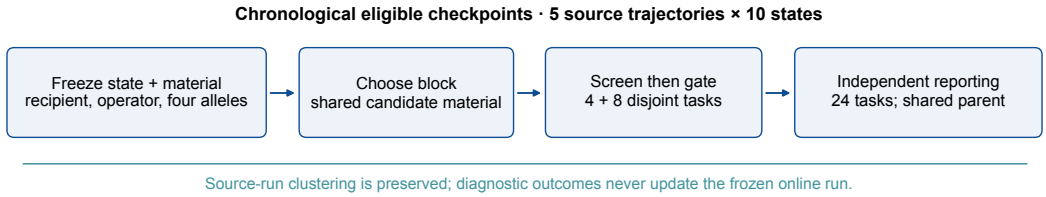


Figure A1: Frozen block-substitution diagnostics. Source states and candidate material are fixed before diagnostic outcomes. Policies select from the same four replacement alleles, then freeze a screen winner, gate disposition, and independently reported child. Shared parent outcomes and source-run clustering are preserved.

The enrollment rule takes the first ten eligible pre-action checkpoints from each of five LinkEvo trajectories, defining 50 states per task. Eligibility requires complete source exposure, valid immutable contracts, and at least two living genomes. It is independent of source-action success, gate gain, and diagnostic reward. Chronological candidate order and exclusions are needed to check this rule; outcome-stratified sequences are not substituted for that order.

Each state fixes the living population, recipient, four replacement alleles and their provenance, operator, environment, archive and cache access, and task/realization keys. Policies share the parent reference, use four screen tasks, eight disjoint gate tasks, and 24 independent reporting tasks, and exclude previously exposed tasks as defined by the diagnostic protocol. Additional proposal work used to prepare diagnostic alleles is charged separately. Diagnostic choices and outcomes never feed back into the frozen online trajectory. The aggregate comparison uses 50 states per task and retains source-run clustering.

B Mechanism controls

B.1 Online component controls

Table 1 isolates signed utility, learned grouping, contextual relevance, and recombination through the definitions in Table 6. Utility neutralization gives 62.00%/52.80% on HotpotQA/HoVer; random hierarchy gives 62.60%/53.60%; context-free history gives 63.26%/54.67%; mutation-only search gives 63.47%/55.13%. LinkEvo reaches 64.40%/56.27%. The differences show that each intervention affects the final population search, with the largest displayed reductions under utility neutralization. Their effects are not summed because actions change subsequent histories and populations.

Table 7: Genetic search behavior pooled over five runs. Gate-passing children undergo population evaluation; retention is a subsequent decision.

Quantity	HotpotQA	HoVer
Completed comparisons	300	287
Mutation comparisons	84	91
Crossover comparisons	216	196
Invalid proposal attempts	8	8
Total action attempts	308	294
Singleton screens	80	75
Multi-locus screens	220	212
Parent screen wins	130	120
Rejected gate winners	100	100
Gate-passing children, evaluated on population panel	70	67
Gate-passing singleton changes	28	24
Gate-passing two-locus changes	28	29
Gate-passing three-locus changes	10	10
Gate-passing four-locus changes	4	4

B.2 Local block-substitution diagnostics

We compare group choices at fixed source states and fixed replacement material. Table 8 uses 50 states and 24 reporting tasks per state. Each of five sources contributes ten states, so equal-source and pooled weighting agree. Net successes subtract shared-parent successes; they are not a separate outcome denominator. LinkEvo improves over the displayed uniform and random-hierarchy group

Table 8: Frozen block-substitution gains on 50 states per task with 24 reporting tasks each. Net successes are child minus shared-parent successes; the denominator is 1,200. Each of five sources contributes ten states.

Local policy	HQ net	HQ gain	HV net	HV gain
LinkEvo contextual groups	44	3.67	49	4.08
Uniform group selection	23	1.92	24	2.00
Random hierarchy	25	2.08	27	2.25
Context-free utility	34	2.83	38	3.17
Full11 local candidate pool	50	4.17	55	4.58
Full16 local candidate pool	66	5.50	68	5.67

selectors. Full11 and Full16 yield larger one-step gains by screening wider pools, which also require more work per decision. The diagnostic fixes operator, recipient, and allele material; it measures the local group choice. Parent wins and rejected children retain the parent and contribute zero update. Negative reporting gains after a passed gate remain included. The local uniform selector acts on fixed material and differs from the complete online uniform-action system.

B.3 Inherited-record and cache access

We separate inherited numerical records from inherited execution outcomes at a common population checkpoint. M controls access to interaction and signed-utility records; C controls exact cached outcomes. Every branch retains literal living genomes and population-panel fitness vectors, pays the work before the checkpoint, and can acquire fresh records and cache entries afterward. The unmodified M1C1 branch reaches 64.40%/56.27%. Hiding inherited records lowers reward to 62.00%/52.60%, while hiding only the inherited cache gives 63.80%/55.67%. The access interaction $K = R_{11} - R_{10} - R_{01} + R_{00}$ is approximately 0.07/0.00 points. This continuation-level interaction is distinct from the instruction-level I and coexists with positive individual record- and cache-access effects.

C Program contracts and resource accounting

Table 9: Inherited-record (M) and execution-cache (C) access at a shared population checkpoint. All branches retain living genomes and their population-panel vectors and pay the common pre-checkpoint acquisition cost. Final rewards are percentages.

Inherited access	HotpotQA	HoVer
M1C1	64.40	56.27
M1C0	63.80	55.66
M0C1	61.59	52.61
M0C0	61.48	52.00

C.1 Program interfaces

Table 10: Retrieval-program contracts. Both programs have four loci; their fourth instruction differs.

Locus	Ordered inputs	Output
First summary	Task input x; first passages with titles	summary1
Second query	x; summary1	query2
Second summary	x; summary1; query2; second passages with titles	summary2
HotpotQA answer	Question x; summary1; summary2	answer
HoVer third query	Claim x; summary1; summary2	query3

The original task input supplies the first retrieval query. HotpotQA retrieves seven passages, produces summary1 and query2, retrieves seven more passages, produces summary2, and extracts an answer. HoVer shares the first three language-model stages, then generates query3 for a final ten-passage retrieval. Its scorer uses the title union across all retrieval stages. Only instruction bodies evolve. Retrieval settings, schemas, parsers, and control flow stay fixed within each cell. Empty demonstrations make these instruction-only programs.

Other program families expose role-specific loci: decomposition, solution, review, and answer extraction for reasoning; specification, plan, code, and revision for coding; and goal-state interpretation, tool selection, argument construction, and state update for tools. A tool locus can execute repeatedly. Turn bounds, revision limits, permitted feedback, and failure parsers stay fixed within each cell. Retriever identity additionally requires corpus and segmentation, index implementation, tokenizer/stemmer/stopwords, rank ties, title mapping, and BM25 parameters.

C.2 Mutation instruction and evidence

The proposer receives the selected recipient's current instructions, immutable contracts, selected loci, and permitted training traces. Evidence includes ordinary initialization and completed screen/gate traces, including recorded execution errors. It excludes population-panel, validation, reporting, and diagnostic traces. Select at most eight most recent eligible traces, breaking completion-time ties by request index. Each identifies source genome, task, module inputs/outputs, and outcome. Whole-trace truncation respects the 6,144-token input envelope. Source labels prevent observations from different genomes being silently attributed to the recipient.

The literal mutation instruction is:

> Revise the instruction bodies at the listed module loci of this fixed program. The input contains current instructions, immutable input/output contracts, and permitted training traces. Return one complete, changed, nonempty instruction for each listed locus in a JSON object with exactly those keys. Coordinate the instructions where one module supplies information another consumes. Preserve all interfaces, tool permissions, and control-flow rules. Do not modify other loci, insert task-specific answers, or use evidence outside the supplied training traces. Each body must satisfy the stated token limit. Return only the JSON object.

The canonical retrieval-locus keys are `first_summary`, `second_query`, `second_summary`, and `answer` or `third_query`. A response supplies exactly the selected subset. Every value must be a complete changed body, nonempty and at most 1,024 tokens under the fixed local tokenizer. The

response allowance is 8,192 provider-counted tokens, including reasoning where applicable. Provider usage and local body-length units remain distinct.

For a request at first summary and second query, the following object satisfies the key and nonempty-body schema; acceptance also checks both bodies against their recipient versions and token limits:

```
{
  "first_summary": "Preserve named entities and unresolved links in the
    retrieved evidence.",
  "second_query": "Use the intermediate entity and unresolved relation to
     target the missing evidence."
}
```

Adding an unrequested `answer` key makes that response invalid, as does returning either body unchanged. The attempt remains charged. Numerical history is read directly; rendering affects proposal input and never determines admission of numerical records. Matched genetic controls use the same evidence categories. Native feedback-policy differences for external baselines belong to their configuration disclosure.

C.3 Cache, duplicate, and failure semantics

Execution-cache identity comprises the full genome, task content, declared environment, and realization key. Environment includes available model and serving revisions, program contracts, tool/retrieval configuration, scorer, and execution settings. A hit references one acquired outcome and its usage provenance. Client-visible protocol identity defines reuse when a hosted backend does not expose immutable internals. Population-panel tasks use fixed per-task realization keys within a run, allowing exact reuse for identical genomes.

Comparison deduplication is separate from execution caching. Its key includes executed arms, operator, block and partition, environment, and screen/gate request keys. An exact replay adds no record, age refresh, or numerical weight. Partly overlapping comparisons can still share outcomes and remain dependent. A duplicate genome never gains another population slot.

Invalid mutation attempts consume a proposal call, action attempt, and available usage without a comparison record or free fallback. Malformed task output receives the declared failure reward with incurred usage retained. Infrastructure termination returns incomplete status and partial costs. Missing token usage propagates an unknown total and makes token-based allocation incomplete. Run status distinguishes launches, completion, invalid proposals, task failures, infrastructure failures, and exclusions. The 8/7 invalid proposals in the retrieval summary are included in all proposal-call totals.

C.4 Action reservations and measured usage

Initialization charges eight ordinary episodes and 24 Default population evaluations. Final selection reserves two programs on 300 validation tasks, leaving

$$B_{\text{search}} = 2232 - 32 - 600 = 1600. \tag{6}$$

The fixed 24-task population panel is a training component and is charged whenever a child must be scored. A child can pass its gate and then be removed during bounded retention. Singleton actions reserve $8 + 16 + 24 = 48$ episodes and multi-locus actions reserve $16 + 16 + 24 = 56$. Full11 and Full16 reserve 84 and 104. These worst-case physical allowances ignore future cache hits. Unused allowance is released; conservative reservation can leave some cache-affordable work unperformed near the cap.

For completed outcome requests,

$$N_{\text{physical}} = N_{\text{logical}} - N_{\text{cache hits}}, \tag{7}$$

$$N_{\text{model}} = \sum_{e \in \text{executed episodes}} n_{\text{calls}}(e) + N_{\text{proposal attempts}} + N_{\text{other calls}}. \tag{8}$$

Incomplete transport attempts and their charged usage are separate entries. Processed tokens sum measured input and output usage across executor, proposer, and other model roles. Reasoning tokens already included in output are counted once. Provider prompt-cache usage is distinct from exact

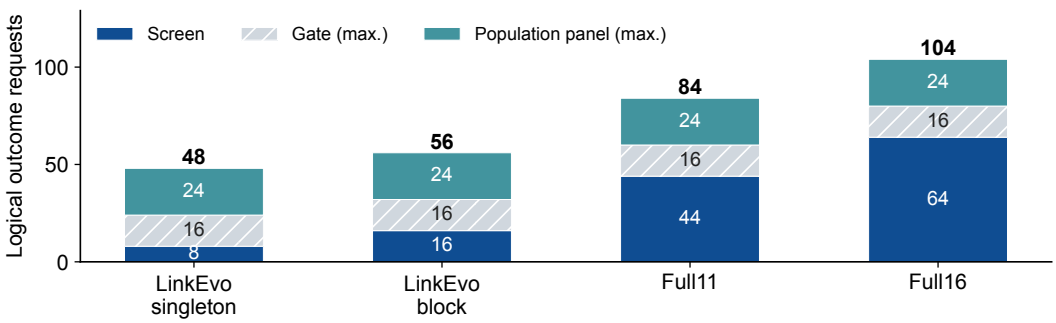


Figure A2: Maximum logical outcome requests before cache reuse. Screening, conditional gating, and conditional population evaluation are separate segments. Parent wins skip later stages; failed gates skip population evaluation. These bars describe analytical action costs, not end-to-end speedups or model-call totals.

program-execution reuse. Numerical control adds no model calls; CPU work belongs in runtime measurements when reported.

Every physical retrieval episode in Table 2 completes four executor calls, so its rows satisfy $N_{\text{model}} = 4N_{\text{physical}} + N_{\text{proposal attempts}}$. This equality is specific to those executions; arbitrary tool loops have episode-dependent call counts. No constant token allowance substitutes for measured consumption.

Table 11: LinkEvo stage totals pooled over five retrieval runs. Subtracting cache hits from logical search requests gives physical search episodes.

Stage / quantity	HotpotQA	HoVer
Screen logical requests	4,160	3,992
Gate logical requests	2,720	2,672
Child population-panel requests	1,680	1,608
Total search logical requests	8,560	8,272
Search cache hits	700	396
Search physical episodes	7,860	7,876
Initialization physical episodes	160	160
Final-selection physical episodes	3,000	3,000
Total physical episodes	11,020	11,036
Proposal attempts	92	98

The search-stage sums give 7,860/7,876 physical episodes. Adding 160 initialization and 3,000 final-selection episodes yields 11,020/11,036 over five runs, or 2,204.0/2,207.2 per run. The corresponding proposal averages are $(84 + 8)/5 = 18.4$ and $(91 + 7)/5 = 19.6$. Per-run episode counts are 2,192, 2,214, 2,206, 2,198, 2,210 for HotpotQA and 2,218, 2,188, 2,216, 2,204, 2,210 for HoVer. Every run respects the cap and leaves less than a 48-episode action reservation.

Table 12: Separately capped runs underlying Figure 2. Rewards are percentages and tokens are measured millions, averaged over five runs. The horizontal episode coordinates are assigned caps.

Cap	Task	Method	Reward	Tokens (M)
816	HotpotQA	LINKEVO	54.88	8.80
816	HotpotQA	Uniform	54.79	8.90
816	HotpotQA	GEPA+Merge	54.80	9.10
816	HoVer	LINKEVO	44.27	9.20
816	HoVer	Uniform	44.13	9.35
816	HoVer	GEPA+Merge	44.25	9.68
1,152	HotpotQA	LINKEVO	59.20	13.00
1,152	HotpotQA	Uniform	58.73	13.25

Cap	Task	Method	Reward	Tokens (M)
1,152	HotpotQA	GEPA+Merge	59.00	13.66
1,152	HoVer	LINKEVO	49.32	13.50
1,152	HoVer	Uniform	48.40	13.81
1,152	HoVer	GEPA+Merge	49.13	14.35
1,536	HotpotQA	LINKEVO	62.14	18.40
1,536	HotpotQA	Uniform	60.66	18.80
1,536	HotpotQA	GEPA+Merge	61.68	19.34
1,536	HoVer	LINKEVO	53.20	19.10
1,536	HoVer	Uniform	50.31	19.52
1,536	HoVer	GEPA+Merge	52.33	20.32
1,884	HotpotQA	LINKEVO	63.60	24.95
1,884	HotpotQA	Uniform	61.60	24.46
1,884	HotpotQA	GEPA+Merge	62.74	25.26
1,884	HoVer	LINKEVO	55.08	24.90
1,884	HoVer	Uniform	51.87	25.66
1,884	HoVer	GEPA+Merge	53.93	26.54
2,232	HotpotQA	LINKEVO	64.40	29.09
2,232	HotpotQA	Uniform	62.00	29.58
2,232	HotpotQA	GEPA+Merge	63.20	30.36
2,232	HoVer	LINKEVO	56.27	30.24
2,232	HoVer	Uniform	52.60	30.87
2,232	HoVer	GEPA+Merge	54.93	31.92

The budget curves use independent capped search configurations with final selection at each assigned cap. Their episode coordinates are caps; measured tokens form the second horizontal axis. Reporting reward does not select a cap or enter adaptation.

C.5 STUDY-LEVEL EXPENDITURE

Table 13: Distinct expenditure scopes. Numerical totals retain their measurement scope.

Scope	Accounting
One selected program	Adaptation in Table 2; includes final selection
Held-out reporting	Outside adaptation; 300 task executions per retrieval run
Budget sweep	Separate adaptation runs at every cap in Table 12
Frozen diagnostics	Preparation, screens, gates, and independent reporting charged separately
Development/calibration	Separate study overhead; no inherited numerical allowance

A complete usage record identifies task, executor, proposer, method configuration, run, stage, cap, logical requests, cache hits, physical episodes, executed calls, attempted proposals, provider input/cached-input/output/reasoning fields, missing-usage status, and completion status. Dollar conversion additionally requires dated rates and hosting assumptions. This paper compares measured within-cell episode, call, and token totals.

D AGGREGATION AND UNCERTAINTY

D.1 CELL SCORES AND PAIRED CONTRASTS

Within a task–executor–method cell, each run averages reporting rewards according to the benchmark metric and the cell score averages runs. A category-macro metric first averages within category with its declared weights. For the primary Qwen retrieval comparisons, five runs each evaluate the same fixed 300 reporting tasks. If $k_{\pi,s}$ is the number of successes for policy π and run label s,

$$\widehat{R}_\pi = \frac{100}{1500} \sum_s k_{\pi,s}, \qquad \widehat{D}_{\pi,\pi'} = \frac{1}{5} \sum_s \frac{k_{\pi,s} - k_{\pi',s}}{3}. \tag{9}$$

Paired contrasts use common reporting membership and unrounded scores. Shared run labels do not imply matched proprietary-service random draws. The 1,500 outcomes repeat 300 task IDs and are not 1,500 independent task draws.

Table 14: Per-run reporting successes out of 300 tasks. Run labels identify the five optimization runs; totals determine pooled means.

Task / method	11	23	47	71	101	Total
HotpotQA LinkEvo	191	191	193	196	195	966
HotpotQA Uniform	183	184	186	188	189	930
HotpotQA Utility-neutralized	181	189	185	186	189	930
HotpotQA GEPA+Merge	189	186	190	193	190	948
HoVer LinkEvo	163	166	168	173	174	844
HoVer Uniform	156	158	154	158	163	789
HoVer Utility-neutralized	159	158	154	159	162	792
HoVer GEPA+Merge	162	161	162	170	169	824

D.2 Conditional run-level intervals

For paired differences $d_s = (k_{\pi,s} - k_{\pi',s})/3$ in percentage points, we report

$$\bar{d} \pm t_{0.975,4} \frac{\mathrm{sd}(d_s)}{\sqrt{5}}, \qquad t_{0.975,4} = 2.776445. \tag{10}$$

Table 15: Paired run-level differences in percentage points with 95% Student-t intervals, conditional on the fixed reporting tasks and configuration. The unit is the run-level paired difference ($n = 5$).

LinkEvo minus	HotpotQA	HoVer
Uniform	2.400 [2.054, 2.746]	3.667 [2.203, 5.130]
Utility-neutralized	2.400 [1.015, 3.785]	3.467 [1.672, 5.261]
GEPA+Merge	1.200 [0.645, 1.755]	1.333 [0.506, 2.161]

These intervals condition on fixed reporting tasks and the shared configuration. Five runs make the Student-t approximation sensitive to the distribution and independence of run-level differences. Task-sampling, shared-initialization, and adaptive-selection uncertainty require additional treatment. Practical equivalence requires a declared tolerance and suitable evidence.

D.3 Diagnostics and cross-cell summaries

Diagnostic state gains first average within source trajectory and then equally over sources. Shared parent outcomes and repeated source histories create dependence across policies and states. Negative reporting gains remain included after a passed gate. A rejected child is not retrospectively adopted using its reporting result.

The transfer matrices contain 36 task–executor cells each. Their summaries aggregate the displayed paired differences and apply decimal half-up rounding after aggregation. Their positive-cell counts, means, and medians are descriptive summaries, not independent replication or significance counts. Each benchmark receives equal weight, so three retrieval benchmarks collectively receive more weight than any single other task family. Benchmark-specific metrics remain separate in absolute-score reporting.

E Complete-optimizer comparisons

We compare the final program's reward and full adaptation expenditure within a fixed task–executor cell. MIPROv2, GEPA, and GEPA with merging optimize instruction bodies and return programs

with empty demonstration fields. MIPRO includes a published zero-shot variant, and GEPA already optimizes instructions alone; we use instruction-only configurations for both (Opsahl-Ong et al., 2024; Agrawal et al., 2026). The proposer is Qwen3.6-35B-A3B, the returned program preserves the common interfaces and tool permissions, and the physical episode ceiling includes each method's acquisition and final selection.

Table 16: Complete-optimizer configurations in the primary retrieval comparison. All rows optimize instructions only and retain the common task program.

Method	Native search component	Adaptation scope
MIPROv2	Instruction candidate generation and search	No demonstrations in the returned program
GEPA	Reflective mutation and diverse candidate selection	Instruction-only artifact; merge disabled
GEPA+Merge	Reflective mutation and system-aware merging	Instruction-only artifact; merge enabled

Mapping native search to the shared protocol. Native candidate-validation or Pareto-panel data are drawn from the optimization split. The separate 300-task final-selection set is reserved for the frozen training-selected candidate versus Default. MIPROv2 may construct training demonstrations internally to propose instructions even when the returned program has no demonstrations. GEPA uses training trajectories and task feedback for reflection. All such proposal and evaluation work belongs to adaptation accounting. These native feedback policies differ from LinkEvo's restricted proposer-trace pool.

Cross-check against the primary papers. The original MIPRO study evaluates a different executor and retrieval scaffold; its zero-shot results therefore identify a method configuration, not a numerical reference for our Qwen3.6 runs (Opsahl-Ong et al., 2024). GEPA's published Qwen3-8B comparisons use a different executor and a larger rollout allowance. Our local HotpotQA comparison applies the official scorer uniformly, without additional Unicode normalization. The HoVer metric retains the shared three-document all-supporting-title retrieval scope. The local GEPA/merge ordering can differ across executors, as the primary paper's own merge analysis demonstrates (Agrawal et al., 2026). Published scores from different splits, executors, or scaffolds are kept separate from Table 1.

Matched genetic controls separate grouping and allocation from population search. GEPA provides the closest complete comparison for reflective mutation and module-level merging; COBRA-Skills provides context for prioritizing evolving textual candidates (Lu et al., 2026). LinkEvo uses measured block substitutions to inform inheritance and allocation. Uniform, random-hierarchy, and mutation-only controls retain their own names rather than being presented as implementations of other optimizers.

F Complete task–executor comparison grid

Table 17 lists all 36 cells represented in Figure 4, including every negative and tied BBEH difference. It reports paired differences under each benchmark's declared metric. Table 1 provides the absolute rewards for the two primary Qwen retrieval programs.

Table 17: All 36 task–executor comparisons, in percentage points. Δ_U and Δ_G compare LinkEvo with Uniform and GEPA+Merge, respectively. The primary absolute-score anchors are in Table 1.

Task	Executor	Δ_U	Δ_G
HotpotQA	Qwen	+2.40	+1.20
HotpotQA	Gemma	+2.93	+1.74
HotpotQA	DeepSeek	+1.88	+0.95
HotpotQA	GPT	+1.11	+0.56
HotpotQA	Claude	+0.97	+0.50
HotpotQA	Gemini	+1.43	+0.84
HoVer	Qwen	+3.67	+1.33

Task	Executor	Δ_U	Δ_G
HoVer	Gemma	+4.10	+1.94
HoVer	DeepSeek	+2.48	+1.18
HoVer	GPT	+1.60	+0.65
HoVer	Claude	+1.31	+0.53
HoVer	Gemini	+2.05	+0.93
MuSiQue	Qwen	+3.03	+1.64
MuSiQue	Gemma	+3.50	+1.80
MuSiQue	DeepSeek	+2.72	+1.20
MuSiQue	GPT	+1.57	+0.80
MuSiQue	Claude	+1.23	+0.77
MuSiQue	Gemini	+1.80	+1.00
BBEH	Qwen	+0.45	+0.04
BBEH	Gemma	+0.93	+0.20
BBEH	DeepSeek	+0.22	+0.00
BBEH	GPT	-0.17	-0.25
BBEH	Claude	+0.00	-0.18
BBEH	Gemini	+0.36	+0.07
LiveCodeBench	Qwen	+2.13	+0.51
LiveCodeBench	Gemma	+2.53	+1.31
LiveCodeBench	DeepSeek	+1.87	+0.68
LiveCodeBench	GPT	+0.93	+0.97
LiveCodeBench	Claude	+0.80	+0.55
LiveCodeBench	Gemini	+1.22	+0.60
BFCL	Qwen	+2.82	+1.37
BFCL	Gemma	+3.60	+1.53
BFCL	DeepSeek	+2.11	+1.09
BFCL	GPT	+1.33	+0.42
BFCL	Claude	+1.09	+0.53
BFCL	Gemini	+1.38	+0.87

The Qwen retrieval differences agree with the unrounded primary means. The broader grid compares separately optimized programs within each executor environment; history is not transferred between these cells. Six benchmarks and six executors give exactly 36 comparisons per reference method. Full11 and Full16 belong to the compatible four-locus retrieval comparison and are not added to this broader grid.

G Prompts and block inheritance

G.1 Initial instruction bodies

The initial body at each retrieval locus expands its ordered input and output fields literally:

First summary. Given the fields task input, first retrieval passages with titles, produce the field summary1.

Second query. Given the fields task input, summary1, produce the field query2.

Second summary. Given the fields task input, summary1, query2, second retrieval passages with titles, produce the field summary2.

HotpotQA answer. Given the fields question, summary1, summary2, produce the field answer.

HoVer third query. Given the fields claim, summary1, summary2, produce the field query3.

The deployed prompts use these explicit field names. Module signatures and parsers remain immutable.

G.2 A summary-to-query inheritance case

On 100 diagnostic reporting tasks, parent, summary-only, query-only, and joint substitutions have 50, 47, 46, and 64 successes. Hence $I = 0.64 - 0.47 - 0.46 + 0.50 = 0.21$ and

Table 18: Instruction bodies in the two-locus inheritance case. Bold phrases identify the coordination between preserved evidence and the next retrieval query.

Locus	Recipient body	Donor body
First summary	Summarize the passages relevant to the question.	Preserve the **named entities, aliases, and relations** needed to connect the current evidence to a missing hop. Distinguish supported facts from **unresolved links**.
Second query	Write a search query that helps answer the question.	Use the **unresolved link and intermediate entity** from the first summary to target missing evidence. Include its identifying name and relation, and avoid repeating established information.

$J = 0.64 - \max(0.50, 0.47, 0.46) = 0.14$. Moving both bodies preserves a useful summary-to-query dependency; either partial insertion reduces reward. These are reporting-panel contrasts, distinct from the four-task screen and eight-task gate used during optimization.

Table 18 records the recipient and donor texts for the inherited block and its aggregate reporting outcome. The two unchanged loci remain fixed throughout the comparison. This controlled substitution isolates the summary-to-query dependency within the selected recipient context.

H Result provenance and comparison scope

The numerical results refer to the population-based LinkEvo configuration in Section 3. Table 19 states the reporting unit of each analysis, keeping online optimization, local diagnostics, and broad task–executor comparisons distinct. Resource accounting includes population evaluation even for

Table 19: Reporting units across the analyses.

Analysis	Reporting unit
Primary retrieval rewards	Five runs, each on the same 300 held-out tasks, within a task–executor–method cell
Paired intervals	Five run-level differences, conditional on the fixed reporting tasks
Resource comparison	Measured adaptation totals averaged over five retrieval runs
Search behavior	Completed operators, invalid proposals, and gate outcomes pooled over those runs
Budget curves	Five separately capped and validated runs at each budget
Local block diagnostics	Fifty frozen states, ten per source trajectory, with 24 reporting tasks per state
Inherited-access controls	Continuations from a shared living-population checkpoint
Transfer matrices	Thirty-six within-cell optimizer differences against each comparator
Prompt inheritance case	Four substitutions on a separate 100-task reporting panel

discarded children and charges invalid proposal attempts. Final selection evaluates the population-panel champion and immutable Default, followed by reporting of one frozen genome. Gate-passing counts describe admission to population evaluation, while post-reduction survival is a different event. Cross-cell medians and positive counts summarize observed effects without treating shared task/model cells as independent significance tests.

The references establish prior methods and benchmark definitions. Baseline scores in the primary table belong to the declared local comparison; published numbers from other splits or scaffolds are not substituted for them.