SPLICE: Structured Prompt Local Iterative Combinatorial Evolution

Anish Acharya* **Phillip Studans†** **Amit Dhanda** **Sina Amini Niaki**
Ninad Vivek Rao **Vishal Amarlal Khatri** **Brian Verkhovsky**
Amazon Web Services

Abstract

Natural language optimization (NLO) aims to improve an agent by having a model critique and rewrite its natural language policy parameters, e.g., system prompts, tool usage instructions, reasoning directives, few-shot demonstrations. These artifacts are multi-section documents in which some sections encode hard-won *contracts* (safety clauses, tool schemas, output formats) that must not drift, and others are legitimate optimization targets. Existing iterative NLO methods treat the artifact as a monolithic string, producing three recurrent failure modes: unsafe global rewrites, beam collapse onto a single lineage, and unchecked bloat – none addressed by formal guarantees. We introduce SPLICE, a unified NLO framework that poses the optimization recipe as **constrained combinatorial search over an edit graph of block-structured, selectively-mutable states**. Under this framework, existing iterative algorithms instantiate a single *batched iterative search* template or depart from it in a component we can name. Theoretically, we map each failure mode to a matched guarantee: **(i)** per-block mutability preserves contract text verbatim, with a price-of-freezing bound showing the trade-off is nearly free when frozen blocks carry small marginal gains; **(ii)** elitist beam search with diversity slots never lets the leader regress on validation, bounds regression under noisy re-evaluation by an exponential tail, and keeps a competitive candidate from a different parent whenever one exists; **(iii)** a ratio length regularizer comes with an explicit accuracy–length bias bound. Candidates are proposed from dual section-local textual gradients; under a proposer-competence condition whose proxy we measure, edit-graph reachability and a convergence rate bound how quickly improving children arrive. Empirically, across seven public benchmarks under five reproduced protocols, SPLICE wins on four and matches the strongest prior method on three. On a production-style contract stress test it keeps the text of the contract clauses that unconstrained optimizers trade away for reward, and ablations of the loop point to critique and elitist search as the main sources of gain.

1 Introduction

An LLM system's behavior is a policy determined by its weights and its text: system prompts, agent instructions, tool descriptions, and the harness code around the model. With black-box access the weights are fixed, and the text is the only part of the policy that can be optimized. *Natural language optimization* (NLO) improves it by having a model repeatedly critique and rewrite it; the formulation applies to any block-structured text, our experiments optimize prompts, and we loosely call the text the *prompt*. Production prompts are multi-section documents (role framing, demonstrations, safety constraints, tool schemas, output specifications) in one token stream, so an edit to one section can

*Correspondence: achanish@amazon.com
†Work done while at Amazon Web Services.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

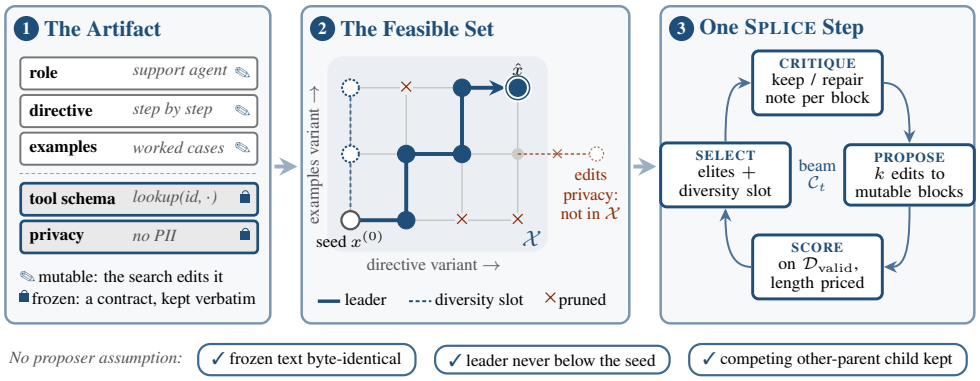


Figure 1: **NLO as constrained combinatorial optimization. (1)** An artifact is a sequence of blocks: mutable blocks are optimization targets, frozen blocks are contracts. **(2)** The feasible set $\mathcal{X}$ is an edit graph; a frozen block admits only its seed text, so editing it is not an edge. Drawn is one SPLICE run: the leader path, a lineage held by a diversity slot, and pruned children. **(3)** One step critiques, proposes k edits to mutable blocks, scores children on $\mathcal{D}_{\text{valid}}$ with length priced, and keeps the elites plus a child of a different parent. Frozen *text* is not frozen behavior (Section 4).

alter behavior governed by another – a **regression risk**. Some sections are *contracts* that must not change, and every added token costs latency: the deployed problem is *constrained*.

Iterative prompt optimization, the most developed form of NLO, follows four directions: *generate-and-select* rewrites [83, 93], *gradient-inspired* refinement from textual feedback [15, 62, 70, 87], *compile-time* module-level search [39, 60], and *evolutionary* population methods [1, 25, 28]. They share two implicit commitments: every part of a candidate is open to editing, and a run is a loop that proposes, scores and keeps candidates. Left implicit, these leave three gaps. *First*, deployment requirements cannot be stated: no prior method exposes *selective mutability* as a property of the search space, and growth is rarely priced ([1] report MIPROv2 prompts $\sim 3\times$ longer at lower accuracy). *Second*, representations, proposals and selectors are confounded across implementations, so methods cannot be compared component by component. *Third*, the search dynamics are untheorized: no monotonicity, no-collapse or length-bias guarantees. The field largely solves the unconstrained relaxation of a constrained problem (Section B provides a detailed related work).

We make both commitments explicit by posing NLO as *constrained combinatorial optimization* (Figure 1). A prompt is a block-structured state $x = (v_1, \ldots, v_m, \pi)$; primitive edits connect states into a finite edit graph; freezing a block is a *feasibility constraint* that makes every edit to it unreachable, and length is a *priced penalty*. This one move closes all three gaps: contracts and cost become part of the problem, existing methods become instances of one batched search template that differ in named components, and search over a finite graph under a noisy oracle admits guarantees whose assumptions can be named. The same propose–score–retain structure underlies self-improvement loops [68, 90, 92], where frozen blocks become invariants the loop cannot rewrite. **SPLICE** (Structured Prompt Local Iterative Combinatorial Evolution) is this framework together with its natural solver (Algorithm 1).

CONTRIBUTIONS.

- **SPLICE, a unified view of NLO as constrained combinatorial optimization** (Section 2), separating representation, proposal mechanism and selector, with contracts as a feasibility constraint and length as a penalty (Definition 11). Existing methods [1, 15, 60, 62, 70, 79, 83, 93] either instantiate a single batched iterative search template (Definition 14) or depart from it in a component we can name: the scoring set, the selector, the output rule, or the expansion (Section C); none makes a section of the prompt unreachable.
- **The SPLICE solver, for the constrained problem** (Section 2). Each production failure mode maps to one design choice: dual section-local critique edits only mutable blocks, a ratio length penalty prices bloat, and elitist selection reserves diversity slots against collapse.
- **A theory of search dynamics** (Section 3). With no assumption on the rewards or the proposer, the **beam leader never regresses** under cached scoring (Lemma 1), the beam keeps a competitive

	Structure	Iterative	DualGrad	SecLocal	Elitism	Diversity	Immutable	Priced	Theory
APE [93]	✗	✗	✗	✗	✗	✗	✗	✗	✗
OPRO [83]	✗	✓	✗	✗	✓	✗	✗	✗	✗
ProTeGi [62]	✗	✓	✗	✗	✗	✗	✗	✗	◐
MAPO [15]	✗	✓	✓	✗	✗	✗	✗	✗	✗
MPO [70]	✓	✓	✗	✓	✗	✗	✗	✗	✗
PO2G [47]	✗	✓	✓	✗	✗	✗	✗	✗	✗
TextGrad [87]	✗	✓	◐	✗	✗	✗	◐	✗	✗
DSPy [39]	✓	✓	✗	✗	✗	✗	✗	✗	✗
MIPROv2 [60]	✓	✓	✗	✗	✗	✗	✗	✗	✗
EvoPrompt [28]	✗	✓	✗	✗	✓	✓	✗	✗	✗
Promptbreeder [25]	✗	✓	✗	✗	✓	✓	✗	✗	✗
GEPA [1]	✓	✓	✗	✗	✓	✓	✗	✗	✗
SPLICE (This Paper)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Representative prompt optimizers against the axes exposed by our framework; Table 5 (Section B) covers every optimizer we discuss or compare against. ✓: the method directly provides the property; ✗: it does not; ◐: partial, or asserted but not established. "DualGrad": positive and negative textual gradients. "SecLocal": updates at the level of named prompt sections. "Elitism": parents can survive a step. "Immutable": designated sections are unreachable by any edit. "Priced": prompt length enters the objective. "Theory": formal guarantees about the search.

candidate from a different parent whenever one exists (Theorem 1), and the quality gate never loosens (Proposition 1). Assuming bounded rewards and i.i.d. batches, if candidates are instead re-scored on n fresh examples each step, the beam's best score **regresses by more than** ε with probability at most $W(1+k)\exp(-n\varepsilon^2/c)$, for a constant $c \leq 2$ (Theorem 2 and Corollary 1). Assuming in addition a proposer whose edits improve with probability at least $p^\star$ from every state ε-far from optimal (Assumption 1), a full beam **proposes an improving child in one step** with probability at least $1-(1-p^\star)^{Wk}$ (Theorem 3), and the search reaches ε-optimality at a rate we give (Theorem 4). Finally, we **price the constraints**: freezing forfeits at most γ^{-1} times the frozen blocks' best single-block gains (Proposition 2), and length regularization biases accuracy by at most $\lambda \cdot \kappa(x^\star)$ (Theorem 5).

- **Empirical validation** (Section 4). On seven public benchmarks (SST-5 [75], GSM8K [14], AIME-2025 [4], ARC-Challenge [13], BBH [76], PUPA [73], GPQA-Diamond [64]) reproducing five published protocols [1, 37, 65, 89, 94] without modification, and calling a margin of at most 0.5 pp a match, SPLICE wins on four and matches the strongest prior method on three (top margin $+5.1$ pp on AIME-2025). On a contract stress test, unconstrained optimizers trade away the clauses SPLICE freezes, yet frozen text is not frozen behavior. Critique lifts Best-of-N by $+1.2$–1.7 pp, and Elitist Beam leads the best plain Best-of-N by $+3.7$ pp on SST-5 and $+1.3$ pp on GPQA (within noise).

2 Structured Local Iterative Combinatorial Evolution

We study Natural Language Optimization (NLO) in the structured, black-box, sample-based regime. A practitioner has a fixed task over an input space $\mathcal{I}$ with unknown distribution $\mathcal{P}$, a black-box LLM (Definition 7), a bounded reward (Definition 8), and an initial prompt (Definition 1) they wish to improve, using an i.i.d. training set $\mathcal{D} \sim \mathcal{P}^N$ to score candidates; disjoint $\mathcal{D}_{\text{valid}}$ and $\mathcal{D}_{\text{test}}$ are reserved for model selection and final reporting, never exposed to the search.

PROMPTS AS BLOCK-STRUCTURED STATES

The prompt is *structured* into semantically coherent blocks that can be rewritten independently; the goal is to pick one variant per block and an admissible ordering so as to maximise expected reward under $\mathcal{P}$ within a finite evaluation budget.

Definition 1 (**Prompt Template**). *A* prompt *consists of* $m \geq 1$ *editable* blocks*, indexed by* $[m] = \{1, \ldots, m\}$*. Each block is a semantically coherent unit whose textual realisation can be varied independently of the others.*

Definition 2 (**Block Variant Sets**). *For each block* $i \in [m]$*, the* variant set $\mathcal{V}_i$ *is a finite non-empty set of admissible textual variants; the initial prompt contributes a distinguished* $v_i^{(0)} \in \mathcal{V}_i$*, so that the initial prompt itself is always feasible.*

Definition 3 (**Block Mutability**). *Let* $I_{\mathrm{mut}} := \{i : |\mathcal{V}_i| \geq 2\}$ *and* $I_{\mathrm{imm}} := [m] \setminus I_{\mathrm{mut}}$ *be the* mutable *and* immutable *block sets. A block* $i \in I_{\mathrm{imm}}$ *has* $\mathcal{V}_i = \{v_i^{(0)}\}$ *and cannot be edited: it encodes a* contract *(safety clause, tool schema, output format) preserved verbatim throughout optimisation.*

Definition 4 (**Block Orderings**). *The* ordering family $\Pi \subseteq \mathfrak{S}_m$ *is a finite family of admissible permutations over* $[m]$*, with distinguished* $\pi^{(0)} \in \Pi$ *corresponding to the initial layout.*

Definition 5 (**Prompt State and Search Space**). *A* prompt state *is a tuple* $x = (v_1, \ldots, v_m, \pi)$ *with* $v_i \in \mathcal{V}_i$ *and* $\pi \in \Pi$*. The* search space *and initial state are*

$$\mathcal{X} = \Big(\prod_{i=1}^{m} \mathcal{V}_i\Big) \times \Pi, \qquad |\mathcal{X}| = |\Pi| \cdot \prod_{i=1}^{m} |\mathcal{V}_i|, \qquad x^{(0)} = (v_1^{(0)}, \ldots, v_m^{(0)}, \pi^{(0)}). \tag{1}$$

$\mathcal{X}$ *is finite but combinatorially large:* $|\mathcal{X}|$ *quickly exceeds any realistic evaluation budget.*

Definition 6 (**Render Function**). *The rendering function* $\mathrm{render} : \mathcal{X} \to \Sigma^*$ *maps a prompt state to a string over alphabet* Σ *(typically Unicode) by concatenating* $v_{\pi(1)}, \ldots, v_{\pi(m)}$ *in the order prescribed by* π *with fixed template separators. It is deterministic and fixed throughout optimisation.*

COMBINATORIAL OBJECTIVE

Definition 7 (**System Agent**). *The system agent is a (possibly stochastic) LLM* $M(\cdot\,; p) : \mathcal{I} \to \mathcal{Y}$ *with* $p \in \Sigma^*$*, accessed strictly as a black box (no gradients, logits, or internal states).*

Definition 8 (**Bounded Reward**). Reward *is a bounded function* $r : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$*; canonical instantiations include* $0/1$ *exact match, accuracy,* F_1*, rubric scores, or rescaled learned-judge outputs.*

Definition 9 (**Population Utility**). *The* population utility *of* $x \in \mathcal{X}$ *is the expected reward under* $\mathcal{P}$*,*

$$F^\star(x) \;:=\; \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{P}}\Big[r\Big(M\big(\mathbf{x};\, \mathrm{render}(x)\big),\, \mathbf{y}\Big)\Big], \tag{2}$$

with expectation over the random example and any stochasticity in M*.*

Because $F^\star$ is inaccessible, the search observes it only through an empirical surrogate:

Definition 10 (**Empirical Utility**). *For* $x \in \mathcal{X}$ *and batch* $B \subseteq \mathcal{D}$*,*

$$F(x;\, B) \;:=\; \frac{1}{|B|} \sum_{(\mathbf{x},\mathbf{y})\in B} r\big(M(\mathbf{x};\, \mathrm{render}(x)),\, \mathbf{y}\big); \tag{3}$$

each oracle call consumes $|B|$ *model evaluations, and we write* $F(x) := F(x;\, \mathcal{D})$*. The gap between* $F^\star$ *(what we optimise) and* F *(what we observe) is the source of the finite-sample bounds in Section E.*

SELECTIVE MUTABILITY AS A FEASIBILITY CONSTRAINT

Definition 3 makes contract preservation a constraint on the feasible set rather than an instruction to the proposer. Each named section of the prompt (role, objective, examples, constraints, tool schema, output format) is tagged `mutable`/`immutable`, and $|\mathcal{V}_i| = 1$ makes every edit to an immutable block unreachable. Freezing shrinks the effective space by $\prod_{i\in I_{\mathrm{imm}}} |\tilde{\mathcal{V}}_i|$, the variants the frozen blocks would otherwise admit, and guarantees the contract's *text* (safety, format, schema) by construction.

LENGTH REGULARISATION AS A SOFT PENALTY

Prompt growth raises latency and cost at every invocation, and unregularised search tends to ratchet length upward when the proposer edits by addition: LLM proposers add content more readily than they remove it [62]. Growth is therefore priced rather than forbidden, with a ratio-based cost

$$\kappa_{\mathrm{len}}(x) := \max\Big(0,\; \tfrac{\mathrm{len}(\mathrm{render}(x))}{\mathrm{len}(\mathrm{render}(x^{(0)}))} - 1\Big). \tag{4}$$

THE CONSTRAINED PROBLEM

Definition 11 (**Constrained Natural Language Optimization**). *Given mutability tags (Definition 3), a cost* $\kappa \geq 0$ *with* $\kappa(x^{(0)}) = 0$*, and a weight* $\lambda \geq 0$*, solve*

$$x_\lambda^\star \;\in\; \arg\max_{x\in\mathcal{X}}\; F_\lambda^\star(x), \qquad F_\lambda^\star(x) \;:=\; F^\star(x) \;-\; \lambda \cdot \kappa_{\mathrm{len}}(x). \tag{5}$$

Setting $\lambda = 0$ and declaring every block mutable recovers the unconstrained problem over the all-mutable space $\mathcal{X}_{\text{all}}$, with optimum $x^\star_{\text{all}}$, which prior methods solve implicitly on a monolithic string.

OPTIMIZATION IS A WALK OVER AN EDIT GRAPH

Problem (5) is combinatorial and F is noisy, so any practical algorithm restricts to *local* moves sampled by a *proposer* and filtered by a *selector*. Prior NLO methods (APE, OPRO, ProTeGi, MAPO, MPO) and SPLICE all fit a single template, differing only in the state space they expose, the signal the proposer consumes, and how the selector retains candidates between rounds.

Definition 12 (Edit Graph). *A* primitive edit *is either a block replacement $v_i \to v_i' \in \mathcal{V}_i \setminus \{v_i\}$ or an admissible reordering $\pi \to \pi' \in \Pi$; each produces a state $x' \in \mathcal{X}$ differing from x in exactly one coordinate. The* edit graph $\mathcal{G} = (\mathcal{X}, E)$ *has radius-one neighbourhood $\mathcal{N}(x) = \{x' : x' \text{ is reachable from } x \text{ by one primitive edit}\}$.*

Replacement and reordering act on disjoint coordinates, so they commute, giving $\operatorname{diam}(\mathcal{G}) \le m_{\text{mut}} + \operatorname{diam}(\Pi)$ with $m_{\text{mut}} = |I_{\text{mut}}|$ (Proposition 3).

Definition 13 (Proposal Distribution). *A* proposal distribution *is a conditional family $q_\phi(\delta \mid x, H_t)$ over primitive edits, supported on edits that produce a feasible state $x' = \text{Apply}(x, \delta) \in \mathcal{X}$, where $H_t = \{(x_s, F(x_s; B_s), c_s)\}_{s<t}$ is the search history. In* SPLICE *and every baseline, q_ϕ is realised by an LLM conditioned on a textual encoding of (x, H_t).*

Feasibility is enforced by rejecting malformed generations (Assumption 2).

Definition 14 (Batched Iterative Prompt Search). *Given q_ϕ, beam width $W \ge 1$, per-parent fan-out $k \ge 1$, and selectors $\{\mathcal{S}_t\}_{t\ge1}$, the iterates are*

$$\mathcal{C}_t = \mathcal{S}_t(\mathcal{F}_t), \quad \mathcal{F}_t = \mathcal{C}_{t-1} \cup \bigcup_{x \in \mathcal{C}_{t-1}} \{\text{Apply}(x, \delta_j) : \delta_j \sim q_\phi(\cdot \mid x, H_t)\}_{j=1}^{k}, \tag{6}$$

starting from $\mathcal{C}_0 = \{x^{(0)}\}$, with $\mathcal{S}_t : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$, $|\mathcal{S}_t(\mathcal{F})| \le W$; $\mathcal{F}_t$ is the frontier.

Greedy hill-climbing, beam search, and elitist beam search recover as specific selectors $\mathcal{S}_t$ on the monolithic state space; Table 1 records where each prior method sits along the four design axes — state space, gradient signal, momentum, selector — that together determine an algorithm (all methods: Table 5). Methods outside the template, such as GEPA, MIPROv2 and OPRO, depart from it in a single named component (Section C, Figure 6). SPLICE's contribution is the particular combination of these axes, which we now describe.

SPLICE realises Definition 14 as a loop (Algorithm 1, Section C). At step t it re-scores the beam $\mathcal{C}_{t-1}$ on a training batch B_t and obtains section-local dual gradients from a critique agent; builds a momentum context from the gradient history G_{t-1}; samples k edits per parent from q_ϕ *restricted to mutable blocks*; scores the children $\mathcal{N}_t$ on $\mathcal{D}_{\text{valid}}$ with $\tilde{F}(x) = F(x; \mathcal{D}_{\text{valid}}) - \sum_j \lambda_j R_j(x)$; keeps W members of $\mathcal{F}_t = \mathcal{C}_{t-1} \cup \mathcal{N}_t$ with an elitist selector that reserves diversity slots; and records (t, g_t, Δ_t) in G_t for the next step's momentum. It returns $\text{leader}(\mathcal{C}_T)$. Selective mutability and the length penalty (above) fix what is optimised; two independent design choices fill the loop's steps, which we take in order.

CRITIQUE: DUAL, SECTION-LOCAL GRADIENTS WITH MOMENTUM

Critique output. The critique agent $\mathcal{G}_{\text{crit}}$ consumes per-example traces on B_t and returns

$$\text{Crit}_t(x) = \big(\mathbf{n}(x),\ \mathbf{p}(x),\ \{(\mathbf{n}_i, \mathbf{p}_i)\}_{i \in I_{\text{mut}}},\ g(x)\big), \tag{7}$$

with failure-patterns $\mathbf{n}$, success-patterns $\mathbf{p}$, per-block assessments $(\mathbf{n}_i, \mathbf{p}_i)$, and a directional summary $g(x)$ ("AWAY from X, TOWARD Y, PRESERVE Z") playing the role of a gradient. Duality keeps the proposer from chasing batch failures while losing working content; section-localisation makes $g(x)$ an addressable per-block update rather than a whole-prompt cue.

Textual momentum. Per-step batches make g_t noisy across steps, and natural-language gradients cannot be averaged vectorially. SPLICE instead surfaces the gradient *trajectory*

$$G_t = \{(s, g_s, \Delta_s)\}_{s \le t}, \quad \Delta_s := \tilde{F}(\text{leader}(\mathcal{C}_s)) - \tilde{F}(\text{leader}(\mathcal{C}_{s-1})), \tag{8}$$

exposing the last w entries ($w = 5$) to the proposer, so it can keep following a direction that has moved the leader. Since the leader never falls (Lemma 1), $\Delta_s \geq 0$: momentum is a record of what helped, not a damper on oscillation.

PROPOSALS AND REACHABILITY

The proposer is conditioned on $\left(\text{render}(x), \tilde{F}(x), \text{Crit}_t(x), G_{t-1}, I_{\text{mut}}\right)$ and samples k structurally-valid edits per parent at non-zero temperature. If the proposer is competent (Assumption 1), the Wk children of a step contain an improving one with probability at least $1 - (1 - p^\star)^{Wk}$ (Theorem 3), so beam width and fan-out trade off directly.

SELECTION: ELITISM WITH DIVERSITY SLOTS

FRONTIER ELITISM. The selector ranks parents and children symmetrically on $\mathcal{F}_t = \mathcal{C}_{t-1} \cup \mathcal{N}_t$ under the regularised validation score $\tilde{F}(\cdot; \mathcal{D}_{\text{valid}})$, computed at first evaluation and *cached* while a candidate stays in the beam (re-rolling would introduce selection-on-noise bias). The leader score $\ell_t := \max_{x \in \mathcal{C}_t} \tilde{F}(x; \mathcal{D}_{\text{valid}})$ therefore never falls (Lemma 1); re-scoring instead would need large batches to control regression (Theorem 2).

DIVERSITY SLOTS. Each candidate carries $\text{id}(x)$, a parent pointer, and an accumulated $\text{lineage}(x)$. SPLICE reserves $D \in \{0, \ldots, W-1\}$ beam slots for diversity, filled by PICKDIVERSE (Algorithm 3): scan the non-elite pool in score order, accept the first candidate clearing the quality gate $\tau_t := r \cdot \ell_t$ under progressively weaker novelty (Tier 1: new lineage born this step; Tier 2: new lineage any generation; Tier 3: different immediate parent), and backfill empty slots. Every ancestor set contains the seed, so in practice Tier 3 operates: the beam keeps a competitive candidate from a different parent whenever one exists (Theorem 1).

THE GATE, AND THE OUTPUT RULE. Monotone ℓ_t (Lemma 1) and $\tau_t = r\ell_t$ give a non-decreasing gate when $r > 0$: a candidate whose cached score falls below τ_t is ineligible for diversity selection thereafter (Proposition 1), which reads as annealing without a temperature (an interpretation, not a guarantee). SPLICE returns $\text{leader}(\mathcal{C}_T)$, whose validation score is biased upward as a maximum over the history (Theorem 7); the reported test score is free of it.

Each design choice is a column of Table 1; Section 3 states the guarantees, and Sections D to E give the full solver, statements and proofs.

3 Theoretical Guarantees

An LLM-driven search does not admit end-to-end guarantees: the proposer is a model we do not control, scores are noisy, and the utility need not be submodular. We therefore state what holds at three levels of assumption: with no assumption on rewards or proposer; with bounded rewards and i.i.d. batches; and, for the *rate* results only, with a competent proposer; Table 6 in Section E lists what each result rests on. Throughout, a candidate's validation score is computed once and *cached* while it stays in the beam. Full statements are in Section E and proofs in Section E.

NO ASSUMPTION ON REWARDS OR PROPOSER

Lemma 1 (Leader monotonicity under cached scoring). *Under cached scoring and the selector $\mathcal{S}^{\text{elit-div}}$, the per-step leader score $\ell_t := \max_{x \in \mathcal{C}_t} \tilde{F}(x; \mathcal{D}_{\text{valid}})$ is non-decreasing: $\ell_t \geq \ell_{t-1}$ $\forall t \geq 1$.*

It is a lemma, not a claim about search dynamics: it follows from the definitions for any elitist selector and any proposer, and the returned prompt therefore never scores below the seed on validation.

Theorem 1 (Diversity preservation). *Let $D \geq 1$ and $r \in [0, 1]$. If $\mathcal{F}_t$ contains $c^\star$ with $\tilde{F}(c^\star) \geq r \cdot \ell_t$ whose parent is not the parent of any elite, then $\mathcal{C}_t$ contains a member scoring at least $r \cdot \ell_t$ whose parent is not the parent of any elite.*

Proposition 1 (Gate monotonicity). *Under cached scoring and $\mathcal{S}^{\text{elit-div}}$ with $r > 0$, the quality floor $\tau_t = r \cdot \ell_t$ is non-decreasing in t, and a candidate whose score falls below τ_t is ineligible for diversity selection at every later step.*

Both guarantees are *conditional*: the selector retains a competitive candidate of a different parent when one exists, and the proposer must generate it (lineage-level variant: Theorem 6).

BOUNDED REWARDS: NOISY NO-REGRESSION

Caching is a choice. If instead every frontier member were re-scored on n fresh i.i.d. examples each step and selected on those scores, regression of the beam's best *population* score $f_t^\star := \max_{x \in \mathcal{C}_t} F^\star(x)$ is controlled by the batch size:

Theorem 2 (Probabilistic no-regression under re-evaluation). *With rewards in $[0,1]$ and an elitist selector, for any $\varepsilon > 0$, $\Pr[f_t^\star < f_{t-1}^\star - \varepsilon] \le (|\mathcal{F}_t| - 1)\exp(-n\varepsilon^2/c)$, where $|\mathcal{F}_t| \le W(1+k)$, and $c = 4\sigma^2 \le 1$ for independent batches and $c = 2$ for a shared batch.*

Corollary 1 (Minimum batch size). $\Pr[f_t^\star < f_{t-1}^\star - \varepsilon] \le \delta$ *whenever* $n \ge (c/\varepsilon^2)\log(W(1+k)/\delta)$.

At W=4, k=3, δ=0.05 and c=1 this requires $n \ge 145$ at $\varepsilon = 0.2$, above most of our per-step batches, which is why SPLICE caches scores instead of re-scoring.

A COMPETENT PROPOSER: RATES

The rate results need one assumption about the LLM proposer, which we cannot prove and instead measure through a proxy (Section F.4.1):

Assumption 1 (Proposal coverage on the ε-sublevel set). *Fix a tolerance $\varepsilon > 0$ in advance. With δ* improving from x *when $\tilde{F}(\mathrm{Apply}(x,\delta)) > \tilde{F}(x)$, let $p^\star(\varepsilon) := \min_{x:\, F^\star(x^\star) - F^\star(x) \ge \varepsilon} \Pr_{\delta \sim q_\phi(\cdot|x,H_t)}[\delta$ is improving from $x]$, uniformly over histories H_t. We assume $p^\star(\varepsilon) > 0$ at this one ε.*

Theorem 3 (Single-step reachability). *Under Assumption 1, at any step at which no beam member is ε-optimal, the step's $|\mathcal{C}_{t-1}|\,k$ proposals contain one that improves on its parent with probability at least $1 - (1 - p^\star(\varepsilon))^{|\mathcal{C}_{t-1}|\,k} =: p^{\mathrm{step}}(\varepsilon)$.*

Theorem 4 (Convergence rate). *Under the scored form of coverage (Assumption 5) at tolerance ε, let $\tilde{h}_0$ be the seed's gap to the scored optimum, $K := \lceil (\tilde{h}_0 - \varepsilon)/\Delta_{\min} \rceil$ for improvement size $\Delta_{\min}$, and $p^{\mathrm{adv}}(\varepsilon)$ a lower bound on the probability that the leader's k proposals improve it by at least $\Delta_{\min}$. Then $T \ge (2K + 8\ln(1/\delta))/p^{\mathrm{adv}}(\varepsilon)$ steps suffice for the leader to be ε-optimal on the scored objective with probability at least $1 - \delta$.*

Either W or k drives the per-step failure probability to zero exponentially. Coverage gates only these two rates, so when it fails the search stalls rather than regresses. Terminal quality also depends on the utility's structure: under γ-submodularity, block-pass greedy is exact when each mutable block contributes one alternative (Proposition 4), and elitist beam on the same draws stays ahead of greedy until they separate (Proposition 5).

THE PRICE OF THE CONSTRAINTS

Proposition 2 (Price of freezing). *Under γ-submodularity of the block-level utility (Definition 16) and diminishing returns on frozen coordinates, with $x^\star_{\mathrm{all}}$ the optimum over the all-mutable space and $x^\star_{\mathrm{mut}}$ the unregularised optimum over $\mathcal{X}$,*

$$F^\star(x^\star_{\mathrm{all}}) - F^\star(x^\star_{\mathrm{mut}}) \;\le\; \frac{1}{\gamma} \sum_{i \in I_{\mathrm{imm}}} \max_{v \in \tilde{\mathcal{V}}_i} \big[F^\star(x^{(0)}[i{:=}v]) - F^\star(x^{(0)})\big]. \tag{9}$$

Theorem 5 (Regularisation bias). *For any $\lambda \ge 0$ and $\kappa \ge 0$, $F^\star(x^\star_{\mathrm{mut}}) - F^\star(x^\star_\lambda) \le \lambda \cdot \kappa(x^\star_{\mathrm{mut}})$.*

We write $x^\star := x^\star_{\mathrm{mut}} = x^\star_\lambda$ at $\lambda = 0$, as in every reported run. Both prices are small when the constraints are used as intended: contract blocks carry little task reward, and λ breaks ties toward shorter prompts. The terminal-quality results rest on γ, for which we have no per-benchmark estimator. The bounds price freezing; they do not show that it is *needed*, nor that it *suffices*, which the contract stress test of Section 4 examines.

4 Experiments

Our experiments address two questions. **(Q1)** Does the batched iterative proposer–scorer template of Definition 14, instantiated with elitist beam search and lineage-aware diversity, translate the deterministic monotonicity of Lemma 1 into measurable gains on standard prompt-optimization benchmarks? **(Q2)** Are those gains robust across target models, task families, and prior-work

Optimizer	ARC	GSM8K
Task Description	79.7	59.4
Expert Prompt	78.4	78.9
Llama Prompt	89.7	79.5
CoT	79.4	76.3
OPRO	79.1	77.1
ProTeGi	78.8	77.3
Evoke	89.0	81.0
EvoPrompt	89.9	81.4
DSPy (MIPROv2, zero-shot)	82.8	77.3
TextGrad	76.5	81.6
UniPrompt + Beam	86.0	82.4
UniPrompt + Greedy	90.5	82.3
DSPy (BFS-RS)	87.5	74.3
DSPy (MIPROv2, few-shot)	86.0	79.7
SPLICE	**91.0**	**83.09**

Table 2: Optimized for model: GPT-3.5-Turbo; datasets: ARC-Challenge, GSM8K [37]. Evoke is Hu et al. [35].

Optimizer	AIME-25	PUPA
Baseline	49.33	78.57
Trace (OptoPrime)	45.33	74.18
MIPROv2 (No-Demos)	48.67	91.85
MIPROv2	51.33	83.37
TextGrad	46.67	85.68
GEPA	59.33	94.47
GEPA + Merge	59.33	**96.46**
SPLICE	**64.44**	96.24

Table 3: Optimized for model: GPT-4.1-Mini; datasets: AIME-2025, PUPA [1]. Trace (OptoPrime) is Cheng et al. [11].

Method	BBH	GPQA-D
Baseline	26.2	65.3
ContraPrompt	89.9	64.3
GEPA	91.9	**68.4**
SPLICE	**92.6**	**68.4**

Table 4: Optimized for model: Claude-4.5 Haiku; datasets: BBH, GPQA-Diamond [65].

evaluation protocols, or are they an artifact of a single setting? To answer both we reproduce, *without modification*, the published protocols of five recent prompt-optimization papers and compare SPLICE against their reported baselines—no re-tuning of their pipelines, no substitution of their target models.

BENCHMARKS.

SETUP. We evaluate SPLICE on seven public benchmarks spanning sentiment classification (SST-5 [75]), mathematical reasoning (GSM8k [14], AIME-2025 [4]), commonsense and multi-step reasoning (ARC-Challenge [13], BBH [76]), instruction following (PUPA [73]), and expert-level science QA (GPQA Diamond [64]). For every benchmark we adopt the target model and evaluation protocol of the corresponding published baseline [1, 37, 65, 89, 94]: GPT-4.1 Mini [1] (AIME-2025, PUPA), GPT-3.5-Turbo [37] (ARC, GSM8k), Claude 4.5 Haiku [65] (BBH, GPQA Diamond), Gemma 3 27B [89] and DeepSeek-V3.2 [94] (SST-5); example optimized prompts are in Section A, further experiments in Section F.

DISCUSSION. Tables 2 to 4 together cover six of the seven benchmarks; the seventh, SST-5, is Table 8 (Section F). Baselines are as published and report no seed variance, so we call a margin of at most 0.5 pp a *match*. On that accounting SPLICE wins on four benchmarks (SST-5, AIME-2025, GSM8K, BBH) and matches the strongest prior method on three (ARC-Challenge at $+0.5$ pp, GPQA-Diamond at 0.0, and PUPA at -0.22 pp against GEPA+Merge). The largest gains are $+5.1$ pp on AIME-2025 over GEPA and $+2.9$ pp on SST-5/`gemma-3-27b`; GSM8K and BBH are narrow ($+0.69$, $+0.70$ pp). Two patterns recur. *First, margins track headroom:* the lead is largest where prior baselines sit well below the task's oracle ceiling and tightens on near-saturated benchmarks (PUPA, BBH); GPQA-Diamond, a tie despite headroom, is the exception. *Second, rank is stable across settings:* on every one of the five target models we tried, SPLICE is at worst within the matching threshold of the best published method, spanning proprietary (GPT-4.1 Mini, GPT-3.5-Turbo, Claude-4.5-Haiku) and open-weight (Gemma-3-27B, DeepSeek-V3.2) tiers.

ABLATIONS.

We study the contribution of each axis of SPLICE in isolation on three public benchmarks. On SST-5 we vary the optimizer LLM across five capacity tiers with two models per tier **M**icro: `gemma-3-4b-it`, `nemotron-nano-9b-v2`; **T**iny: `gemma-3-12b-it`, `nemotron-nano-12b-v2`; **S**mall: `qwen3-32b`, `nemotron-nano-3-30b`; **L**arge: `nemotron-super-3-120b`, `qwen3-vl-235b-a22b`; **F**rontier: `claude-sonnet-4-6`, `claude-opus-4-6`, and on ARC-Challenge across the S, L and F models; on GPQA-Diamond we instead hold the proposer fixed

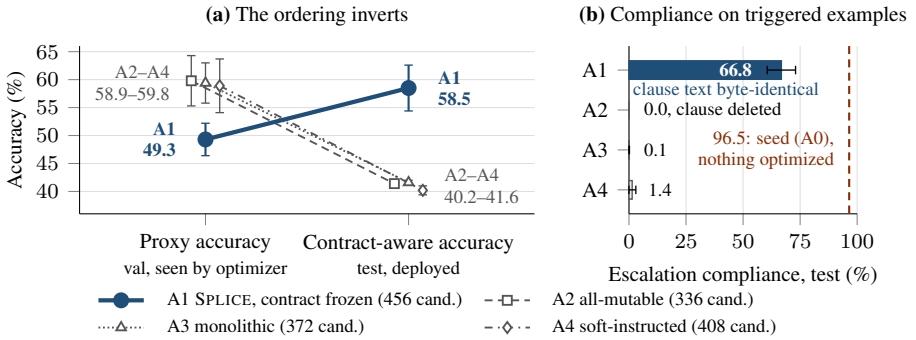
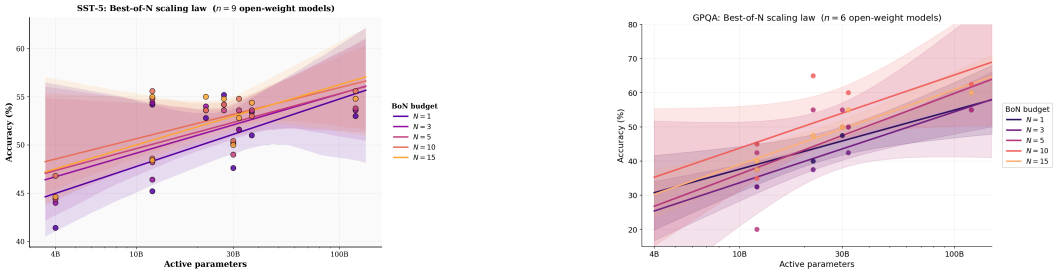


Figure 2: **Under a contract, the ranking inverts between the reward the optimizer sees and the one a deployment pays.** SST-5, `gemma-3-27b-it`, mean ± s.d. over three paired seeds. **(a)** Endpoints are different metrics on different splits; only the ordering is meaningful. **(b)** Only SPLICE keeps the clause in force (full numbers: Table 12, Section F.2).



(a) SST-5, by proposer size

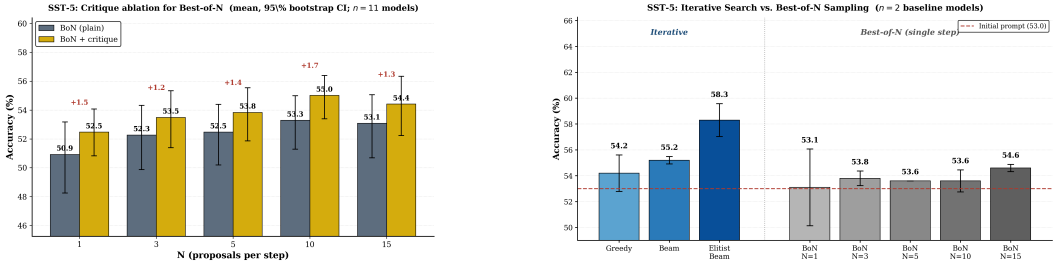
(b) GPQA, by proposer size

Figure 3: **Best-of-N scaling with model size.** Fitted one-shot accuracy rises with model size at every N, while the N-curves overlap within each size band: more samples do not substitute for scale.

(`claude-sonnet-4-6`) and vary the *target* across the S, L and F models. We compare four selectors spanning the iterative / non-iterative divide: Best-of-N (BoN; a single step of Definition 14 that keeps the best of N children and drops the seed), Greedy ($W{=}1$, children only), Beam ($W{>}1$, children only), and Elitist Beam (Algorithm 2), which realises the guarantee of Lemma 1, and that of Theorem 1 when $D \geq 1$ ($D{=}0$ in the GPQA sweep, $D{=}1$ on SST-5).

WHY MUTABILITY: A CONTRACT STRESS TEST. We add a seven-clause contract (tool schema, safety text, output envelope, business rule) to SST-5 on `gemma-3-27b-it`, including POLICY-1: *escalate any snippet mentioning "film" or "movie"*; the proxy reward is accuracy, so obeying it (26% of examples) forfeits reward. The same optimizer (Section 2) runs in four arms that differ *only* in the contract's mutability: A1 frozen ($|\mathcal{V}_i| = 1$), A2 all-mutable, A3 one monolithic editable block, as prior methods search, and A4 editable plus "do not edit". The ordering *inverts* (Figure 2): every editable arm beats A1 by ≈ 10 pp on the proxy and loses by ≈ 17 pp on the contract-aware objective, while the other contract classes were untouched by every arm: optimizers trade away exactly the clauses that cost reward. Freezing is therefore needed, since instruction does not protect the clause, but it does not suffice: it protects text, not effect, and A1's compliance (66.8%) is below the unoptimized seed's 96.5% (details: Section F.2).

SCALING W BEAM WIDTH. We characterize the weakest iterative regime first: Best-of-N sampling with no feedback. Figure 3 reports Best-of-N accuracy across proposer scales and $N \in \{1, 3, 5, 10, 15\}$ on SST-5 and GPQA Diamond. Two facts stand out. First, one-shot accuracy scales strongly with proposer capacity—SST-5 rises from $\sim$45% at the M-tier to $\sim$55% at the L-tier. Second, the N-axis is not monotonic: on SST-5 the five N-curves lie within a narrow band at every scale; on GPQA aggregate accuracy peaks at $N{=}10$ and drops at $N{=}15$, as does the small tier, while the large tier peaks at $N{=}5$ and the frontier tier rises to $N{=}15$, within roughly ± 20 pp of sampling uncertainty. Drawing more samples from the same seed without feedback is not a reliable substitute for capacity or for iteration; beam width is discussed in Section F.3.1.



(a) **Critique** (b) **Iteration**

Figure 4: **Critique and elitist iteration each add accuracy on SST-5.** **(a)** Critique adds 1.2–1.7 pp to Best-of-N at every N: BoN (grey) vs. BoN+critique (gold), per-N gap in red; 95% bootstrap CIs over the 10 optimizer models overlap at every N. **(b)** Elitist search leads one-shot sampling: test accuracy, mean over the two target models; the dashed line is the initial prompt.

ROLE OF CRITIQUE SIGNAL. The next component is the critique signal the appended textual gradient that the proposer conditions on. Figure 4a adds critique to BoN and reports the paired comparison on SST-5, averaged across the ten optimizer models. At every N the critique-on variant outperforms critique-off by $+1.2$ to $+1.7$ pp. Per-arm 95% CIs overlap, so the evidence is the consistent sign across N. The gap is largest at $N{=}10$ ($+1.7$ pp) and does not shrink systematically with N, consistent with textual gradients carrying information that raw sampling does not supply.

GAINS FROM ITERATIVE OPTIMIZATION. Iteration with an elitist selector is what turns critique into durable progress. Figure 4b compares iterative strategies (Greedy, Beam, Elitist Beam) against BoN with the same proposer model on SST-5, and Figure 12 (Section F) on ARC-Challenge. Every iterative strategy clears the initial-prompt baseline on both; BoN also clears it, by less than Elitist Beam, and on the judge task it falls below the seed (Section F.5). Elitist Beam leads the best plain BoN configuration by $+3.7$ pp on SST-5 (averaged over two targets) and by $+1.3$ pp on GPQA Diamond, inside that sweep's noise; on saturated ARC-Challenge the strategies sit within run-to-run variation. Since the proposer is identical across strategies, the gain comes not from proposer capacity but from the scoring–proposal feedback loop with elitist retention (Theorem 3 composed with Lemma 1).

PROPOSER AND TARGET CAPACITY. Fitted one-shot accuracy rises with *proposer* size at every N (Figures 3a and 3b), consistent with Theorem 3. With the proposer fixed and the *target* varied (GPQA), iterative strategies lead on the frontier tier but not on the small tier (Figure 15), since a weak target bounds what any prompt achieves.

SEARCH STRATEGY AND DIVERSITY. At matched $W{=}4$ on BBH, varying only the number of diversity slots, $D{=}0$ stays near 0.21 for all 13 steps (the beam never escapes a single mode) while every $D \geq 1$ variant exceeds 0.92 by step 4 (Table 13, Section F); elsewhere the knob is far weaker, so diversity slots are insurance against collapse rather than a uniform gain (search trees: Figure 13, Section F). Figure 14 (Section F) traces validation score along each run's best-path lineage: on frontier targets the best paths gain $+20$ to $+22.5$ pp over the seed in three to four nodes, while L1 needs six nodes to cover a $+47.5$ pp gap ($27.5\% \to 75\%$); the leader is non-decreasing along the best path (Lemma 1) and climbs most in the first two steps (Section F.4.1).

5 Limitations and Conclusion

Limitations. The guarantees are conditional. Proposal coverage gates only the rate results, so when it fails the search stalls rather than regresses; the terminal-quality results rest on the submodularity ratio γ, for which we have no estimator. Freezing preserves a contract's text, not its effect: in Section 4 the frozen clause survives in every candidate, yet behavioral compliance on test falls from 96.5% for the unoptimized seed to 66.8%, so frozen blocks still need a behavioral audit. SPLICE optimizes the reward it is given, so a misspecified reward is the dominant failure mode. Open problems: which neighboring edits erode a frozen clause, an estimator for γ, and a matched-budget cost comparison.
Conclusion. Casting prompt optimisation as constrained search over an edit graph of selectively-mutable blocks, SPLICE carries guarantees with named assumptions and wins on four of seven benchmarks, matching on three. The open case is recursive: the optimized text is the optimizer's own.

Acknowledgments and Disclosure of Funding

We thank Benjamin Wyant and Matt Yanchyshyn for their leadership and support of this work, and the anonymous reviewers and area chair, whose detailed feedback significantly improved the quality of this paper. This work was funded by Amazon Web Services, where all authors are employed; no third-party funding was received, and the authors declare no other competing interests.

References

[1] L. A. Agrawal, S. Tan, D. Soylu, N. Ziems, R. Khare, K. Opsahl-Ong, A. Singhvi, H. Shandilya, M. J. Ryan, M. Jiang, C. Potts, K. Sen, A. Dimakis, I. Stoica, D. Klein, M. Zaharia, and O. Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=RQm2KQTM5r`.

[2] J.-Y. Audibert and S. Bubeck. Best arm identification in multi-armed bandits. In *Conference on Learning Theory (COLT)*, 2010.

[3] D. Austin and E. Chartock. GRAD-SUM: Leveraging gradient summarization for optimal prompt engineering. *arXiv preprint arXiv:2407.12865*, 2024.

[4] M. Balunović, J. Dekoninck, I. Petrov, N. Jovanović, and M. Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

[5] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[6] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

[7] A. Blum and M. Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1006–1014, 2015.

[8] B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[10] L. Chen, M. Feldman, and A. Karbasi. Weakly submodular maximization beyond cardinality constraints: Does randomization help greedy? *arXiv preprint arXiv:1707.04347*, 2017.

[11] C.-A. Cheng, A. Nie, and A. Swaminathan. Trace is the next AutoDiff: Generative optimization with rich feedback, execution traces, and LLMs. *arXiv preprint arXiv:2406.16218*, 2024.

[12] J. Cheng, X. Liu, K. Zheng, P. Ke, H. Wang, Y. Dong, J. Tang, and M. Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL `https://arxiv.org/abs/2311.04155`.

[13] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[14] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[15] A. Cui, P. Nandyalam, A. Rufail, E. Cheung, A. Lei, K. Zhu, and S. O'Brien. Introducing MAPO: Momentum-aided gradient descent prompt optimization. *arXiv preprint arXiv:2410.19499*, 2024.

[16] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

[17] M. Davari, U. Garg, W. Cai, and E. Belilovsky. Stabilizing black-box prompt optimization with textual regularization and signal aggregation, 2025. URL `https://arxiv.org/abs/2507.09839`.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[19] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. P. Xing, and Z. Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[20] V. Y. Dong, R. Pryzant, Y. Liu, and J. Jiao. CAPO: Constraint-aware prompt optimization for LLM agents, 2026. URL `https://arxiv.org/abs/2608.16068`.

[21] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1–2):51–81, 2002.

[22] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 117–126, 2015.

[23] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[24] E. Even-Dar, S. Mannor, Y. Mansour, and S. Mahadevan. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(6), 2006.

[25] C. Fernando, D. S. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *International Conference on Machine Learning*, pages 13481–13544. PMLR, 2024.

[26] F. Glover. Tabu search — part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[27] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *International Conference on Genetic Algorithms (ICGA)*, pages 41–49, 1987.

[28] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*, 2023.

[29] Y. Han, Y. Han, S. Huang, G. Liu, Z. Zhou, B. Liu, Y. Zhang, I. N. Shi, L. He, and T. Shi. MAPGD: Multi-agent prompt gradient descent for collaborative prompt optimization, 2025. URL `https://arxiv.org/abs/2509.11361`.

[30] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[31] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[32] O. Honovich, U. Shaham, S. R. Bowman, and O. Levy. Instruction induction: From few examples to natural language task descriptions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

[33] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2004.

[34] S. Hu, C. Lu, and J. Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

[35] X. Hu, P. Tang, S. Zuo, Z. Wang, B. Song, Q. Lou, J. Jiao, and D. Charles. Evoke: Evoking critical thinking abilities in LLMs via reviewer-author prompt editing. *arXiv preprint arXiv:2310.13855*, 2023.

[36] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization (LION)*, pages 507–523, 2011.

[37] G. Juneja, G. Jajoo, H. Li, J. Jiao, N. Natarajan, and A. Sharma. Task facet learning: A structured approach to prompt optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23473–23496, 2025.

[38] E. Kaufmann, O. Cappé, and A. Garivier. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17(1):1–42, 2016.

[39] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, et al. DSPy: compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2023.

[40] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[41] A. Krause and D. Golovin. Submodular function maximization. In *Tractability*, pages 71–104. 2014.

[42] N. Kulin, V. Zhuravlev, A. Khairullin, S. Muravyov, I. Makarov, D. Sukhorukov, and E. Averkova. From monolithic to modular: Segment-level automatic prompt optimization. In *IJCAI-ECAI Workshop on Robustifying Generative AI (RobustifAI)*, 2026. URL `https://arxiv.org/abs/2608.11219`.

[43] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011.

[44] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[45] D. Li, C. Shi, W. Ou, C. Shen, and J. Yang. Efficient multi-objective prompt optimization via pure-exploration bandits. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://arxiv.org/abs/2605.14553`.

[46] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[47] A. Lieander, J. H. Wang, and K. Rafferty. Prompt optimization with two gradients for classification in large language models. *AI*, 6(8):182, 2025.

[48] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[49] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[50] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

[51] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9), 2023.

[52] B. T. Lowerre. *The harpy speech recognition system.* Carnegie Mellon University, 1976.

[53] P. Lu, Z. Hong, Z. Shang, Z. Wang, Y. Ban, Y. Shu, M. Zhang, S. Qiu, and Z. Dai. FedPOB: Sample-efficient federated prompt optimization via bandits, 2025. URL `https://arxiv.org/abs/2509.24701`.

[54] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[55] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, and P. Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[56] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[57] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[58] J. Nian, Z. Peng, H. Shang, D. H. Park, and Y. Fang. Submodular evaluation subset selection in automatic prompt optimization, 2026. URL `https://arxiv.org/abs/2601.03493`.

[59] A. Novikov, N. Vũ, M. Eisenberger, E. Dupont, P.-S. Huang, A. Z. Wagner, S. Shirobokov, B. Kozlovskii, F. J. R. Ruiz, A. Mehrabian, M. P. Kumar, A. See, S. Chaudhuri, G. Holland, A. Davies, S. Nowozin, P. Kohli, and M. Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[60] K. Opsahl-Ong, M. J. Ryan, J. Purtell, D. Broman, C. Potts, M. Zaharia, and O. Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, 2024.

[61] A. Prasad, P. Hase, X. Zhou, and M. Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *European Chapter of the Association for Computational Linguistics (EACL)*, 2023.

[62] R. Pryzant, D. Iter, J. Li, Y. Lee, C. Zhu, and M. Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7957–7968, 2023.

[63] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, 2019.

[64] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[65] R. Rishav, P. Pujari, and P. Rastogi. Contraprompt: Contrastive prompt optimization via dyadic reasoning trace analysis. In *ICLR 2026 Workshop on Logical Reasoning of Large Language Models*, 2026. URL `https://openreview.net/forum?id=86E9P53xbM`.

[66] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. R. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi, P. Kohli, and A. Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[67] D. Russo and J. Zou. Controlling bias in adaptive data analysis using information theory. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1232–1240, 2016.

[68] J. Schmidhuber. Goedel machines: Self-referential universal problem solvers making provably optimal self-improvements. *arXiv preprint arXiv:cs/0309048*, 2003.

[69] X. Sécheresse, J.-Y. Guilbert-Ly, and A. Villedieu de Torcy. GAAPO: Genetic algorithmic applied to prompt optimization. *arXiv preprint arXiv:2504.07157*, 2025.

[70] P. Sharma and A. Z. Henley. Modular prompt optimization: Optimizing structured prompts with section-local textual gradients. *arXiv preprint arXiv:2601.04055*, 2026.

[71] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[72] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[73] L. Siyan, V. C. Raghuram, O. Khattab, J. Hirschberg, and Z. Yu. PAPILLON: Privacy preservation from Internet-based and local language model ensembles. In L. Chiruzzo, A. Ritter, and L. Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3371–3390, Albuquerque, New Mexico, Apr. 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.173. URL https://aclanthology.org/2025.naacl-long.173/.

[74] C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[75] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[76] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, A. Kluska, A. Lewkowycz, A. Agarwal, A. Power, A. Ray, A. Warstadt, A. W. Kocurek, A. Safaya, A. Tazarv, A. Xiang, A. Parrish, A. Nie, A. Hussain, A. Askell, A. Dsouza, A. Slone, A. Rahane, A. S. Iyer, A. J. Andreassen, A. Madotto, A. Santilli, A. Stuhlmüller, A. M. Dai, A. La, A. K. Lampinen, A. Zou, A. Jiang, A. Chen, A. Vuong, A. Gupta, A. Gottardi, A. Norelli, A. Venkatesh, A. Gholamidavoodi, A. Tabassum, A. Menezes, A. Kirubarajan, A. Mullokandov, A. Sabharwal, A. Herrick, A. Efrat, A. Erdem, A. Karakaş, B. R. Roberts, B. S. Loe, B. Zoph, B. Bojanowski, B. Özyurt, B. Hedayatnia, B. Neyshabur, B. Inden, B. Stein, B. Ekmekci, B. Y. Lin, B. Howald, B. Orinion, C. Diao, C. Dour, C. Stinson, C. Argueta, C. Ferri, C. Singh, C. Rathkopf, C. Meng, C. Baral, C. Wu, C. Callison-Burch, C. Waites, C. Voigt, C. D. Manning, C. Potts, C. Ramirez, C. E. Rivera, C. Siro, C. Raffel, C. Ashcraft, C. Garbacea, D. Sileo, D. Garrette, D. Hendrycks, D. Kilman, D. Roth, C. D. Freeman, D. Khashabi, D. Levy, D. M. González, D. Perszyk, D. Hernandez, D. Chen, D. Ippolito, D. Gilboa, D. Dohan, D. Drakard, D. Jurgens, D. Datta, D. Ganguli, D. Emelin, D. Kleyko, D. Yuret, D. Chen, D. Tam, D. Hupkes, D. Misra, D. Buzan, D. C. Mollo, D. Yang, D.-H. Lee, D. Schrader, E. Shutova, E. D. Cubuk, E. Segal, E. Hagerman, E. Barnes, E. Donoway, E. Pavlick, E. Rodolà, E. Lam, E. Chu, E. Tang, E. Erdem, E. Chang, E. A. Chi, E. Dyer, E. Jerzak, E. Kim, E. E. Manyasi, E. Zheltonozhskii, F. Xia, F. Siar, F. Martínez-Plumed, F. Happé, F. Chollet, F. Rong, G. Mishra, G. I. Winata, G. de Melo, G. Kruszewski, G. Parascandolo, G. Mariani, G. X. Wang, G. Jaimovitch-Lopez, G. Betz, G. Gur-Ari, H. Galijasevic, H. Kim, H. Rashkin, H. Hajishirzi, H. Mehta, H. Bogar, H. F. A. Shevlin, H. Schuetze, H. Yakura, H. Zhang, H. M. Wong, I. Ng, I. Noble, J. Jumelet, J. Geissinger, J. Kernion, J. Hilton, J. Lee, J. F. Fisac, J. B. Simon, J. Koppel, J. Zheng, J. Zou, J. Kocon, J. Thompson, J. Wingfield, J. Kaplan, J. Radom, J. Sohl-Dickstein, J. Phang, J. Wei, J. Yosinski, J. Novikova, J. Bosscher, J. Marsh, J. Kim, J. Taal, J. Engel, J. Alabi, J. Xu, J. Song, J. Tang, J. Waweru, J. Burden, J. Miller, J. U. Balis, J. Batchelder, J. Berant, J. Frohberg, J. Rozen, J. Hernandez-Orallo, J. Boudeman, J. Guerr, J. Jones, J. B. Tenenbaum, J. S. Rule, J. Chua, K. Kanclerz, K. Livescu, K. Krauth, K. Gopalakrishnan, K. Ignatyeva, K. Markert, K. Dhole, K. Gimpel, K. Omondi, K. W. Mathewson, K. Chiafullo, K. Shkaruta, K. Shridhar, K. McDonell, K. Richardson, L. Reynolds, L. Gao, L. Zhang, L. Dugan, L. Qin, L. Contreras-Ochando, L.-P. Morency, L. Moschella, L. Lam, L. Noble, L. Schmidt, L. He, L. Oliveros-Colón, L. Metz, L. K. Senel, M. Bosma, M. Sap, M. T. Hoeve, M. Farooqi, M. Faruqui, M. Mazeika, M. Baturan, M. Marelli, M. Maru, M. J. Ramirez-Quintana, M. Tolkiehn, M. Giulianelli, M. Lewis, M. Potthast, M. L. Leavitt, M. Hagen, M. Schubert, M. O. Baitemirova, M. Arnaud, M. McElrath, M. A. Yee, M. Cohen, M. Gu, M. Ivanitskiy, M. Starritt, M. Strube, M. Swędrowski, M. Bevilacqua, M. Yasunaga, M. Kale, M. Cain, M. Xu, M. Suzgun, M. Walker, M. Tiwari, M. Bansal, M. Aminnaseri, M. Geva, M. Gheini, M. V. T, N. Peng, N. A. Chi, N. Lee, N. G.-A. Krakover, N. Cameron, N. Roberts, N. Doiron, N. Martinez, N. Nangia, N. Deckers, N. Muennighoff, N. S. Keskar, N. S. Iyer,

N. Constant, N. Fiedel, N. Wen, O. Zhang, O. Agha, O. Elbaghdadi, O. Levy, O. Evans, P. A. M. Casares, P. Doshi, P. Fung, P. P. Liang, P. Vicol, P. Alipoormolabashi, P. Liao, P. Liang, P. W. Chang, P. Eckersley, P. M. Htut, P. Hwang, P. Miłkowski, P. Patil, P. Pezeshkpour, P. Oli, Q. Mei, Q. Lyu, Q. Chen, R. Banjade, R. E. Rudolph, R. Gabriel, R. Habacker, R. Risco, R. Millière, R. Garg, R. Barnes, R. A. Saurous, R. Arakawa, R. Raymaekers, R. Frank, R. Sikand, R. Novak, R. Sitelew, R. L. Bras, R. Liu, R. Jacobs, R. Zhang, R. Salakhutdinov, R. A. Chi, S. R. Lee, R. Stovall, R. Teehan, R. Yang, S. Singh, S. M. Mohammad, S. Anand, S. Dillavou, S. Shleifer, S. Wiseman, S. Gruetter, S. R. Bowman, S. S. Schoenholz, S. Han, S. Kwatra, S. A. Rous, S. Ghazarian, S. Ghosh, S. Casey, S. Bischoff, S. Gehrmann, S. Schuster, S. Sadeghi, S. Hamdan, S. Zhou, S. Srivastava, S. Shi, S. Singh, S. Asaadi, S. S. Gu, S. Pachchigar, S. Toshniwal, S. Upadhyay, S. S. Debnath, S. Shakeri, S. Thormeyer, S. Melzi, S. Reddy, S. P. Makini, S.-H. Lee, S. Torene, S. Hatwar, S. Dehaene, S. Divic, S. Ermon, S. Biderman, S. Lin, S. Prasad, S. Piantadosi, S. Shieber, S. Misherghi, S. Kiritchenko, S. Mishra, T. Linzen, T. Schuster, T. Li, T. Yu, T. Ali, T. Hashimoto, T.-L. Wu, T. Desbordes, T. Rothschild, T. Phan, T. Wang, T. Nkinyili, T. Schick, T. Kornev, T. Tunduny, T. Gerstenberg, T. Chang, T. Neeraj, T. Khot, T. Shultz, U. Shaham, V. Misra, V. Demberg, V. Nyamai, V. Raunak, V. V. Ramasesh, vinay uday prabhu, V. Padmakumar, V. Srikumar, W. Fedus, W. Saunders, W. Zhang, W. Vossen, X. Ren, X. Tong, X. Zhao, X. Wu, X. Shen, Y. Yaghoobzadeh, Y. Lakretz, Y. Song, Y. Bahri, Y. Choi, Y. Yang, S. Hao, Y. Chen, Y. Belinkov, Y. Hou, Y. Hou, Y. Bai, Z. Seid, Z. Zhao, Z. Wang, Z. J. Wang, Z. Wang, and Z. Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=uyTL5Bvosj`. Featured Certification.

[77] X. Wang and T. Wang. FOZO: Forward-only zeroth-order prompt optimization for test-time adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL `https://arxiv.org/abs/2603.04733`.

[78] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[79] X. Wang, C. Li, Z. Wang, F. Bai, H. Luo, J. Zhang, N. Jojic, E. P. Xing, and Z. Hu. PromptAgent: Strategic planning with language models enables expert-level prompt optimization. In *International Conference on Learning Representations (ICLR)*, 2024.

[80] Z. Wang, Y. Dong, O. Delalleau, J. Zeng, G. Shen, D. Egert, J. J. Zhang, M. N. Sreedhar, and O. Kuchaiev. HelpSteer2: Open-source dataset for training top-performing reward models. *Advances in Neural Information Processing Systems*, 37:1474–1501, 2024.

[81] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[82] Y. Wu, Y. Gao, B. B. Zhu, Z. Zhou, X. Sun, S. Yang, J.-G. Lou, Z. Ding, and L. Yang. StraGo: Harnessing strategic guidance for prompt optimization, 2024. URL `https://arxiv.org/abs/2410.08601`.

[83] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=Bb4VGOWELI`.

[84] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[85] L. Yin and Z. Wang. LLM-AutoDiff: Auto-differentiate any LLM workflow. *arXiv preprint arXiv:2501.16673*, 2025.

[86] X. Yin, X. Wang, L. Pan, L. Lin, X. Wan, and W. Y. Wang. Gödel agent: A self-referential agent framework for recursive self-improvement. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. arXiv:2410.04444.

[87] M. Yuksekgonul, F. Bianchi, J. Boen, S. Liu, Z. Huang, C. Guestrin, and J. Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

[88] T. Zehle, M. Schlager, T. Heiß, and M. Feurer. CAPO: Cost-aware prompt optimization. In *Proceedings of the Fourth International Conference on Automated Machine Learning*, volume 293 of *Proceedings of Machine Learning Research*, pages 18/1–45. PMLR, 2025.

[89] T. Zehle, T. Heiß, M. Schlager, M. Aßenmacher, and M. Feurer. promptolution: A unified, modular framework for prompt optimization. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 282–296, 2026.

[90] E. Zelikman, E. Lorch, L. Mackey, and A. T. Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2310.02304.

[91] J. Zhang, J. Xiang, Z. Yu, F. Teng, X. Chen, J. Chen, M. Zhuge, X. Cheng, S. Hong, J. Wang, B. Zheng, B. Liu, Y. Luo, and C. Wu. AFlow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.

[92] J. Zhang, S. Hu, C. Lu, R. Lange, and J. Clune. Darwin Godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

[93] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023.

[94] Z. Zhou, Y. Wu, J. Yang, Z. Xiao, and R. Li. Evaluating the effectiveness of black-box prompt optimization as the scale of LLMs continues to grow. *arXiv preprint arXiv:2505.08303*, 2025.

[95] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Appendix

Contents

A Example Prompts Generated **18**

B Related Work **26**

B.1 Natural Language Optimization 26

B.2 Optimization Foundations 28

C Framework Details **29**

C.1 Algorithms as Selectors 29

D Solver Details **30**

D.1 Proposals 30

E Theory: Supporting Results and Proofs **31**

E.1 Probabilistic Preliminaries 33

E.2 Edit-Graph Reachability 36

E.3 Approximate Submodularity and Greedy Approximation 39

E.4 Elitist Monotonicity and Noisy No-Regression 43

E.5 Diversity Preservation and the Quality Gate 48

E.6 Generalization, Regularization, and the Price of Freezing 50

F Experimental Details **54**

F.1 Protocol and Reporting 54

F.2 Does the Constrained Formulation Buy Anything? 56

F.3 Knobs, and Where They Fail 59

F.4 When Iteration Pays: Capacity, $\hat{p}^\star$, and Cost 61

F.5 Application: LLM-as-Judge Rubric Calibration 63

F.6 Additional Search Strategy Ablations. 65

F.7 GPQA Diamond: Search-Tree Topology 65

F.8 Candidate Lineage and Score Progression on SST-5 71

A Example Prompts Generated

Seed Prompt for AIME

```
<role></role>
<objective>Solve the given math problem. Every Answer is a non-negative integer between 0 and 999
(inclusive).</objective>
<approach></approach>
<examples></examples>
```

<output_format>give the final answer in the form \boxed{N}, where N is an integer between 0 and 999.</output_format>

SPLICE's Optimized Prompt for AIME, GPT-4.1 Mini

<role>You are a world-class competition mathematician with expertise across all domains: number theory, combinatorics, algebra, geometry, probability, and analysis. You are known for your meticulous calculations, creative problem-solving approaches, and systematic verification. You never rush — you break complex problems into manageable pieces and double-check each step before proceeding.</role>

<objective>Solve the given math competition problem. Produce a clear, rigorous, step-by-step solution. The final answer is always a non-negative integer between 0 and 999 (inclusive). Before stating your final answer, verify it through an independent check or sanity test.</objective>

<approach>Use this disciplined problem-solving framework:

1. **Parse the problem precisely**: Identify exactly what is being asked. Note whether the answer requires a modular reduction (e.g., mod 1000, mod 1000000007), a specific quantity, or a transformed result. Misreading the question is the #1 source of errors.
2. **Identify the mathematical domain and select tools**: Determine the core techniques needed (modular arithmetic, CRT, generating functions, PIE, Heron's formula, coordinate geometry, recursion, casework, etc.).
3. **Break into sub-problems**: Decompose complex problems into smaller, independently verifiable steps. For each sub-step, compute carefully and verify before moving on.
4. **Execute with extreme arithmetic care**:
   - For modular arithmetic: reduce intermediate results frequently to keep numbers small.
   - For counting: clearly define what you're counting, check for over/undercounting.
   - For algebra: expand and simplify carefully, checking each manipulation.
   - For geometry: verify formulas before applying, check units/dimensions.
5. **Verify your answer**: Use at least one of these checks:
   - Compute via an alternative method
   - Check small cases or boundary conditions
   - Verify the answer is in the valid range [0, 999]
   - For mod problems, confirm the reduction was applied correctly
   - Estimate the magnitude to see if the answer is reasonable

Critical pitfalls to avoid:

- Forgetting to apply the final modular reduction
- Off-by-one errors in counting (inclusive vs exclusive bounds)
- Sign errors in algebraic manipulation
- Miscounting cases in casework
- Confusing permutations with combinations
- Forgetting to account for symmetry or identical elements

</approach>

<examples>
Example 1:
Input: Find the remainder when 2^{100} is divided by 1000.
Output: We need $2^{100} \mod 1000$.
Since $1000 = 8 \times 125$, we use CRT.
$2^{100} \mod 8 = 0$ (since $2^3 = 8$ divides 2^{100}).
$2^{100} \mod 125$: Using repeated squaring.
$2^{10} = 1024 \equiv 24 \pmod{125}$
$2^{20} \equiv 24^2 = 576 \equiv 76 \pmod{125}$
$2^{40} \equiv 76^2 = 5776 \equiv 26 \pmod{125}$
$2^{80} \equiv 26^2 = 676 \equiv 51 \pmod{125}$
$2^{100} = 2^{80} \times 2^{20} \equiv 51 \times 76 = 3876 \equiv 1 \pmod{125}$

CRT: $x \equiv 0 \pmod 8$, $x \equiv 1 \pmod{125}$.
$x = 125k + 1$, need $125k + 1 \equiv 0 \pmod 8 \to 5k \equiv 7 \pmod 8 \to k \equiv 3 \pmod 8$.
$x = 125(3) + 1 = 376$.
$\boxed{376}$

Example 2:
Input: How many ways can you arrange the letters in BANANA?
Output: BANANA has 6 letters: B(1), A(3), N(2).
Distinct arrangements $= 6!/(1! \times 3! \times 2!) = 720/12 = 60$.
$\boxed{60}$

Example 3:
Input: In triangle ABC, $AB = 13$, $BC = 14$, $CA = 15$. Find the area of triangle ABC.
Output: Using Heron's formula with $s = (13 + 14 + 15)/2 = 21$.
Area $= \sqrt{21 \times 8 \times 7 \times 6} = \sqrt{7056} = 84$.
Verification: With base $BC = 14$, height $h = 2(84)/14 = 12$. Check: $13^2 = 5^2 + 12^2 = 169$ ✓
$\boxed{84}$

Example 4:
Input: A fair six-sided die is rolled 4 times. What is the probability that the sum is 10? Express your answer as p/q in lowest terms and find $p + q$.
Output: We need the number of solutions to $d_1 + d_2 + d_3 + d_4 = 10$ where $1 \le d_i \le 6$.
Substitute $x_i = d_i - 1$, so $x_1 + x_2 + x_3 + x_4 = 6$ where $0 \le x_i \le 5$.
Using inclusion-exclusion on stars and bars:

- No restriction: $\binom{9}{3} = 84$
- One $x_i \ge 6$: $\binom{4}{1}\binom{3}{3} = 4$
- Two $x_i \ge 6$: impossible (sum would be $\ge 12 > 6$)

Favorable $= 84 - 4 = 80$.
Total outcomes $= 6^4 = 1296$.
Probability $= 80/1296 = 5/81$.
$p + q = 5 + 81 = 86$.
$\boxed{86}$

Example 5:
Input: Find the number of integers n with $1 \le n \le 1000$ such that $n^2 + n$ is divisible by 6.
Output: $n^2 + n = n(n + 1)$, the product of two consecutive integers.
$n(n + 1)$ is always even (divisible by 2), so we need $n(n + 1)$ divisible by 3.
$n(n + 1) \bmod 3$: if $n \equiv 0$: $0 \times 1 \equiv 0$ ✓; if $n \equiv 1$: $1 \times 2 \equiv 2$ ×; if $n \equiv 2$: $2 \times 0 \equiv 0$ ✓.
So $n(n + 1)$ is NOT divisible by 3 when $n \equiv 1 \pmod 3$.
Numbers 1–1000 with $n \equiv 1 \pmod 3$: these are $1, 4, 7, \ldots, 1000$. Count $= (1000 - 1)/3 + 1 = 334$.
Answer $= 1000 - 334 = 666$.
$\boxed{666}$
</examples>

<output_format>give the final answer in the form \boxed{N}, where N is an integer between 0 and 999.</output_format>

Seed Prompt for GSM8K

<role></role>
<objective>Solve the given grade school math word problem step by step, showing your reasoning, and provide the final numerical answer.</objective>
<approach></approach>
<examples></examples>
<output_format>Show your step-by-step reasoning, then on the final line output the answer as a single number after '#### '. For example: '#### 42'. Do not include units, commas, or any text after the number.</output_format>

SPLICE's Optimized Prompt for GSM8K, GPT-3.5-Turbo

<role>You are a precise math word problem solver. You work through problems using plain-text arithmetic only. CRITICAL FORMAT RULES: Never use LaTeX, backslash-parentheses \(...\), dollar signs $...$,

\boxed{}, or any markup notation anywhere in your response. Write all calculations as plain text like: 3 $\times$ 4 = 12. Always produce a final whole number answer (no decimals, no trailing zeros, no commas in numbers).</role>

<objective>Solve the given math word problem. Before computing, carefully identify: (1) what quantity is being asked for, (2) any real-world facts needed but not stated (e.g., cars have 4 wheels, roller skates have 4 wheels each, a pair = 2 items, a dozen = 12), and (3) the correct direction of comparisons (e.g., "A costs twice as much as B" means A = 2 $\times$ B). Then solve step by step using plain arithmetic. Output ONLY a bare integer after #### on the final line.</objective>

<approach>Follow these steps:

1. **Read the problem twice**. Identify the exact question being asked. Note all units carefully — distinguish between groups vs individuals, per-person vs total, containers vs contents. When N people/items are excluded from an action, determine what unit that removes (e.g., "2 students didn't eat" removes 2 complete servings, not 2 individual pieces).
2. **State any real-world knowledge needed**: common objects' properties (wheels on vehicles, legs on animals, items in a pair/dozen), and any implicit facts the problem assumes you know.
3. **Parse comparative relationships carefully**. "A is/costs twice as much as B" means A = 2 $\times$ B (A is larger). "A is 5 less than 3 times B" means A = 3 $\times$ B $-$ 5. Identify the base quantity and the derived quantity before computing.
4. **Compute step by step** using plain-text arithmetic ($\times$, $+$, $-$, $/$). Calculate each intermediate value as a number before proceeding. Never leave expressions unevaluated. Never use variables, algebra, or any LaTeX/markup.
5. **Verify**: Does your answer match the exact question asked? Did you account for ALL items/people mentioned? Is it a whole number? Write it after "#### " on the last line.

</approach>

<examples>
Example 1:
Input: A store had 120 shirts. They sold 1/3 of them on Monday and 1/4 of the remainder on Tuesday. How many shirts are left?
Output: The store started with 120 shirts. Sold on Monday: 120 $\times$ 1/3 = 40 shirts. Remaining after Monday: 120 $-$ 40 = 80 shirts. Sold on Tuesday: 80 $\times$ 1/4 = 20 shirts. Remaining after Tuesday: 80 $-$ 20 = 60 shirts.
60

Example 2:
Input: 5 cars are in a parking lot. How many wheels are there in total?
Output: There are 5 cars in the parking lot. Each car has 4 wheels (real-world fact). Total wheels: 5 $\times$ 4 = 20 wheels.
20

Example 3:
Input: A plot of land costs $40,000. Construction costs twice as much as the land. What is the total cost of land plus construction?
Output: Land cost: 40000 dollars. Construction costs twice as much as land: 2 $\times$ 40000 = 80000 dollars. Total cost: 40000 + 80000 = 120000 dollars.
120000

Example 4:
Input: A teacher bought 10 packs of crackers, with 8 crackers per pack, for her class. 2 students were absent and didn't get any. How many crackers were eaten?
Output: Total packs: 10. Each pack has 8 crackers. Total crackers: 10 $\times$ 8 = 80 crackers. 2 students were absent, so 2 packs were not eaten. Crackers not eaten: 2 $\times$ 8 = 16 crackers. Crackers eaten: 80 $-$ 16 = 64 crackers.
64

Example 5:
Input: A worker earns $12 per hour. She works 8 hours a day for 5 days. Her employer deducts $50 for insurance. What is her take-home pay?
Output: Hourly wage: 12 dollars. Daily earnings: 12 $\times$ 8 = 96 dollars. Weekly earnings: 96 $\times$ 5 = 480 dollars. After insurance deduction: 480 $-$ 50 = 430 dollars.
430
</examples>

<output_format>Show your step-by-step reasoning, then on the final line output the answer as a single number after '#### '. For example: '#### 42'. Do not include units, commas, or any text after the number.</output_format>

Seed Prompt for GPQA

<role>You are an expert scientist with deep knowledge in physics, chemistry, and biology. You approach problems methodically by identifying the relevant scientific principle first, then systematically evaluating each option.</role>

<objective>Answer the following multiple choice question by reasoning through it step by step, then selecting the correct answer.</objective>

<approach></approach>

<examples></examples>

<output_format>Think through the problem step by step. After your reasoning, state your final answer on a new line in the format: The answer is (X) where X is A, B, C, or D.</output_format>

SPLICE's Optimized Prompt for GPQA, Claude 4.5 Haiku

<role>You are an expert-level science examiner who solves graduate-level multiple choice questions with precision and efficiency. Your expertise spans physics, chemistry, biology, and mathematics including specialized topics like quantum mechanics, organic synthesis, molecular biology, optics, thermodynamics, and statistical mechanics. You are decisive: you commit to a solution path, execute it cleanly, and state your answer. You never ramble or restart calculations unnecessarily.</role>

<objective>Answer the multiple choice question below. Be concise and decisive. You MUST always end with your final answer in the required format — never leave it unstated, even if your reasoning feels incomplete.</objective>

<approach>Use this efficient reasoning strategy:

1. **CLASSIFY**: What type of problem is this?
2. **ELIMINATE**: Rule out obviously wrong options.
3. **SOLVE**: Apply the relevant principle ONCE. Do not restart.
4. **MATCH & COMMIT**: Match your result to the closest option.
5. **ANSWER**: State your final answer immediately.

CRITICAL RULES:

- Keep reasoning to under 15 sentences.
- If you catch yourself restarting, STOP and commit.
- The final answer line is MANDATORY.

</approach>

<examples>
Example 1:
Input: What is the ground state energy of a quantum harmonic oscillator with angular frequency ω? (A) 0 (B) $\hbar\omega/2$ (C) $\hbar\omega$ (D) $3\hbar\omega/2$
Output: CLASSIFY: Conceptual recall — standard quantum mechanics formula. ELIMINATE: (A) is wrong — zero-point energy is nonzero. (D) is the energy of $n = 1$ state. SOLVE: Ground state ($n = 0$) has $E = \hbar\omega(n + 1/2) = \hbar\omega/2$. MATCH: This is option (B).
The answer is (B)

Example 2:
Input: A 2 kg block slides down a frictionless incline of 30° for 3 meters. What is its speed at the bottom? (A) 3.8 m/s (B) 5.4 m/s (C) 7.7 m/s (D) 9.8 m/s
Output: CLASSIFY: Calculation — energy conservation. ELIMINATE: (D) equals g, unlikely for this setup. SOLVE: Height $= 3 \times \sin 30^\circ = 1.5$ m. $v = \sqrt{2 \times 9.8 \times 1.5} \approx 5.42$ m/s. MATCH: Option (B).
The answer is (B)
</examples>

<output_format>Think through the problem step by step. After your reasoning, state your final answer on a new line in the format: The answer is (X) where X is A, B, C, or D.</output_format>

Seed Prompt for BBH

<role></role>
<objective>Solve the reasoning task. Think step by step, then give the final answer.</objective>
<approach></approach>
<examples></examples>
<output_format>Think through the problem step by step. After your reasoning, state your final answer on a new line starting with: The answer is</output_format>

SPLICE's Optimized Prompt for BBH, Claude 4.5 Haiku

<role>You are a precise reasoning engine. You solve problems step by step using plain text only — no markdown, no bold, no backticks, no special formatting. Your final line is always "The answer is" followed by the answer value. You never write anything after that final line.</role>

<objective>Solve the given reasoning task using step-by-step plain-text reasoning, then state your final answer.</objective>

<approach>Follow these steps:

1. Read the problem carefully. Think through it step by step in plain text.
2. Determine the correct answer. For pronoun resolution or ambiguity questions, choose the most natural and contextually supported interpretation.
3. Format your answer value according to the question type:
   - Multiple choice: lowercase letter in parentheses, e.g. (c)
   - Yes/no or true/false: just the word
   - Sorting or listing: items separated by spaces
   - Completion: only the missing part
   - Open-ended: the shortest correct answer
4. End with exactly one line: "The answer is" followed by your answer value.
5. STOP after that line. Do NOT repeat or write anything after it.

</approach>

<examples>
Example 1 (Multiple choice):
Input: Which of the following is a fruit? Options: (a) carrot (b) apple (c) potato (d) spinach
Output: Apples are fruits; carrots, potatoes, and spinach are vegetables.
The answer is (b)

Example 2 (Yes/no):
Input: If all roses are flowers and some flowers fade quickly, does it follow that some roses fade quickly? Answer yes or no.
Output: The flowers that fade quickly may not be roses. The conclusion does not follow from the premises.
The answer is no

Example 3 (Sorting):
Input: Sort the following words alphabetically: pear apple orange banana
Output: Alphabetical order: apple, banana, orange, pear.
The answer is apple banana orange pear

Example 4 (Completion):
Input: Complete the sequence: 2, 4, 8, 16, __
Output: Each term is doubled. 16 times 2 equals 32.
The answer is 32

Example 5 (Reasoning / pronoun resolution):
Input: The trophy doesn't fit into the brown suitcase because it is too large. What does "it" refer to? Options: (a) the trophy (b) the suitcase
Output: If something does not fit because it is too large, the thing that is too large is the object trying to fit in, which is the trophy.
The answer is (a)

Example 6 (True/false):
Input: Is the following statement true or false: "The sum of the interior angles of a triangle is 180 degrees."
Output: In Euclidean geometry, the interior angles of any triangle sum to 180 degrees.

The answer is true
</examples>

<output_format>Think through the problem step by step. After your reasoning, state your final answer on a new line starting with: The answer is</output_format>

Seed Prompt for ARC

<role>You have to solve the following science question.</role>

<objective>Analyze the science question, reason through all answer choices, and give your final answer inside <Answer></Answer> tags containing ONLY the letter A, B, C, or D.</objective>

<approach></approach>

<examples>
Example 1:
Input: What is the boiling point of water at standard atmospheric pressure?
A) 50°C B) 100°C C) 150°C D) 200°C
Output: Water boils at 100 degrees Celsius at standard atmospheric pressure.
<Answer>B</Answer>
</examples>

[Immutable suffix appended to every prompt]: After any reasoning, give your final answer inside <Answer></Answer> tags containing ONLY the letter A, B, C, or D.

SPLICE's Optimized Prompt for ARC, GPT-3.5-Turbo

<role>You are a precise science reasoning expert. You read questions carefully, evaluate every answer choice against exact scientific definitions, and always select the answer that most directly addresses what the question asks — whether that answer is simple or highly technical. You do not let the framing or sophistication of an option override its accuracy.</role>

<objective>Analyze the science question and evaluate ALL answer choices before selecting your final answer. Consider why each option is correct or incorrect. When options seem equally plausible, use precise scientific definitions to distinguish them. Give your final answer inside <Answer></Answer> tags containing ONLY the letter A, B, C, or D.</objective>

<approach>Use this 5-step strategy:

1. **CLASSIFY** the question type — this determines your reasoning lens:
   - **CAUSAL/CONSEQUENCE**: "what happens when X?" or "what is the result of X?" → trace the MOST DIRECT causal chain from X to the effect; do not skip to secondary or downstream consequences.
   - **MECHANISM/APPEARANCE**: "why does X appear/seem...?" → focus on the perceptual or observational mechanism, not intrinsic physical properties.
   - **PROPERTY/IDENTIFICATION**: "which property identifies X?" or "what state of matter is X?" → match precise scientific definitions to the property described.
   - **STRUCTURAL/LAYER**: "which layer does X?" → verify the spatial relationship (above/below/adjacent) explicitly.
2. **IDENTIFY** the core scientific concept and the exact qualifier being tested (e.g., "regardless of size," "appears," "most direct," "solid," "primarily").
3. **EVALUATE** each answer choice (A, B, C, D) with a brief justification — explain why each is correct or incorrect using precise scientific definitions. Do not stop at the first plausible option.
4. **CHALLENGE** — Compare your top 2 candidates:
   (a) EXACT MATCH: Does this answer address what the question literally asks, or does it answer a related-but-different question?
   (b) CAUSAL DIRECTNESS: For causal questions, which answer is the most immediate consequence of the stated cause — not a secondary effect?
   (c) REFERENT CHECK: Confirm which entity, layer, or process the question names. Make sure your answer describes THAT entity, not an adjacent one.
5. **SELECT** the answer that best matches the question's exact ask. If you are uncertain or running low on space, skip extended reasoning and immediately output your best answer in <Answer> tags.

</approach>

<examples>
Example 1 (Causal):
Input: A large forest is cut down for lumber. What is the MOST DIRECT consequence for local wildlife?
A) Global carbon dioxide levels increase significantly
B) Animals lose their habitat and food sources
C) Soil erosion increases over many decades
D) Nearby rivers become warmer due to less shade
Output: Question type: CAUSAL/CONSEQUENCE — asks for the most direct consequence of forest removal for wildlife.
A) CO_2 increase — a real effect of deforestation, but it operates at a global scale and is an atmospheric consequence, not a direct consequence FOR local wildlife.
B) Animals lose habitat and food sources — forest removal directly eliminates the shelter and food that wildlife depend on. This is the immediate, first-order effect on wildlife.
C) Soil erosion — a real long-term consequence, but it operates over decades and affects the land, not wildlife directly.
D) Warmer rivers — a secondary physical effect; it may eventually affect aquatic life, but it is not the most direct consequence of forest removal for wildlife generally.
Challenge (top 2: A vs B): A describes a global atmospheric consequence. B describes the direct, immediate effect on the animals themselves. The question asks for the most direct consequence FOR local wildlife — B is the precise match.
Causal directness check: Forest removed $\rightarrow$ habitat and food gone $\rightarrow$ wildlife directly affected. B is the first-order link in the chain.
<Answer>B</Answer>

Example 2 (Mechanism):
Input: Frost forms on a cold window on a winter morning. Which process best describes how the frost formed?
A) Condensation, as water vapor cools below the dew point
B) Freezing, as liquid water droplets on the glass solidify overnight
C) Deposition, as water vapor converts directly to ice without passing through a liquid phase
D) Sublimation, as ice on nearby surfaces vaporizes and reattaches to the glass
Output: Question type: MECHANISM — asks which phase-change process produces frost.
A) Condensation — gas to liquid phase change (produces dew, not frost). Only applies above 0°C.
B) Freezing — liquid to solid. Requires liquid water already present; does not describe frost forming from air.
C) Deposition — direct gas to solid phase change (water vapor $\rightarrow$ ice), bypassing liquid entirely. This is exactly how frost forms on surfaces below 0°C.
D) Sublimation — solid to gas (the reverse process). Describes ice disappearing, not forming.
Challenge (top 2: A vs C): Both involve water vapor, but condensation produces liquid (dew, above freezing) while deposition produces solid ice (frost, below freezing). The window is below 0°C — deposition is the exact mechanism.
Exact match check: The question asks which process describes how frost forms $\rightarrow$ deposition directly answers this.
<Answer>C</Answer>

Example 3 (Appearance):
Input: Venus appears brighter than any other planet in the night sky as seen from Earth. What is the primary reason Venus appears so bright?
A) Venus reflects a large amount of sunlight due to its thick cloud cover
B) Venus is the planet closest to the Sun
C) Venus has a very large diameter compared to other planets
D) Venus emits its own light from volcanic activity
Output: Question type: MECHANISM/APPEARANCE — asks why Venus APPEARS bright from Earth, not what its intrinsic physical property is.
A) High reflectivity from thick clouds — Venus has an albedo of $\sim$0.65–0.70, reflecting most incoming sunlight. High reflectivity directly causes exceptional apparent brightness as seen from Earth.
B) Closest to the Sun — Mercury is even closer to the Sun. More importantly, orbital proximity to the Sun does not determine how bright a planet appears from Earth; reflectivity and distance to Earth do.
C) Large diameter — Jupiter and Saturn are far larger than Venus but appear dimmer. Size alone does not determine apparent brightness.
D) Emits own light — planets reflect sunlight; they do not generate visible light. Volcanic emission is negligible.
Challenge (top 2: A vs B): B answers "which planet is near the Sun" — a related but different question. A

directly answers why Venus APPEARS bright from Earth through the reflectivity mechanism.
Exact match check: Appearance → reflectivity mechanism → A.
<Answer>A</Answer>

Example 4 (Structural):
Input: Which layer of the Earth moves on top of the asthenosphere and is responsible for tectonic plate movement?
A) Inner core B) Outer core C) Lithosphere D) Mesosphere
Output: Question type: STRUCTURAL/LAYER — asks which layer sits ON TOP OF (above) the asthenosphere.
A) Inner core — Earth's deepest solid layer; does not interact with the asthenosphere.
B) Outer core — liquid iron-nickel layer deep inside Earth; does not overlie the asthenosphere.
C) Lithosphere — the rigid outer shell (crust + uppermost mantle) that sits directly on top of the semi-plastic asthenosphere and is divided into tectonic plates.
D) Mesosphere (lower mantle) — lies BELOW the asthenosphere, not above it.
Challenge (top 2: C vs D): The asthenosphere is the viscous middle layer. Lithosphere is above it; mesosphere is below. The question asks for the layer moving ON TOP — that is the lithosphere.
Referent check: "moves on top of the asthenosphere" → above → lithosphere = C.
<Answer>C</Answer>
</examples>

[Immutable suffix appended to every prompt]: After any reasoning, give your final answer inside <Answer></Answer> tags containing ONLY the letter A, B, C, or D.

B Related Work

We review NLO approaches, from prompt optimization, its most developed form, to the self-improvement loops and test-time search that share its structure (Section B.1), and then the optimization results our analysis builds on (Section B.2). Table 5 locates each NLO method along the axes of Definition 14.

B.1 Natural Language Optimization

Discrete search. The text that steers a model matters: prompt quality drives few-shot performance [9], the right directive elicits stepwise reasoning [81], and accuracy is sensitive even to the order of prompt parts [54]; we study black-box optimization of discrete text [51], leaving aside soft prompts that require gradient access [44, 46]. Early methods searched tokens or phrases by gradients [71], a learned policy [19] or edit operators [61], with feedback that does not describe what failed. APE [93] and instruction induction [32] let an LLM propose whole instructions and keep the best, a single step with a greedy selector in the terms of Definition 14.

Textual feedback. ProTeGi [62] introduced the *textual gradient*, a natural-language account of minibatch errors that drives a beam over edits; its beam keeps only new children, so a strong parent can be lost. Successors enrich the signal: score histories [83], positive gradients and momentum [15, 47], feedback from correct predictions to curb drift [17, 82], contrasted reasoning traces [65], author–reviewer loops [35], aggregated feedback [3], and tree search over edit sequences [79]. StraGo [82] names the failure we target, a revision that fixes current errors but breaks passing examples, and answers it with better guidance rather than a selection rule. Feedback and search have also been applied to whole programs: DSPy [39] and MIPROv2 [60] compile per-module prompts by searching over instruction and demonstration candidates, while TextGrad [87], LLM-AutoDiff [85] and Trace [11] back-propagate textual feedback through computation graphs. These systems optimize every prompt in a program; they structure the pipeline, whereas we structure the interior of one artifact, so the two compose. TextGrad and AdalFlow can freeze a whole variable, but not a section inside one. Any of these feedback operators could serve as our proposer q_ϕ.

Evolutionary and reflective search. EvoPrompt [28], Promptbreeder [25] and GAAPO [69] evolve prompt populations with LLM operators. GEPA [1], the strongest published baseline on four of our seven benchmarks (counting GEPA+Merge on PUPA), mutates by reflecting on execution traces and draws parents from a Pareto front over per-instance scores. That front is the closest prior mechanism to our diversity slots, but it is defined over task instances rather than parentage, so children of one

Table 5: **Every prompt optimizer we discuss or compare against, on the axes of our framework** (Definition 14), grouped as in Section B.1. ✓: the method provides the property; ✗: it does not; ◐: partial, or asserted but not established. Column definitions and per-method notes follow the table (Section B.2).

	Structure	Iterative	DualGrad	SecLocal	Elitism	Diversity	Immutable	Priced	Theory
Generate-and-select									
APE [93]	✗	✗	✗	✗	✗	✗	✗	✗	✗
Instruction induction [32]	✗	✗	✗	✗	✗	✗	✗	✗	✗
AutoPrompt [71]	✗	✓	✗	✗	✗	✗	✗	✗	✗
RLPrompt [19]	✗	✓	✗	✗	✗	✗	✗	✗	✗
GrIPS [61]	✗	✓	✗	✗	✗	✗	✗	✗	✗
Iterative with textual feedback									
ProTeGi [62]	✗	✓	✗	✗	✗	✗	✗	✗	◐
OPRO [83]	✗	✓	✗	✗	✓	✗	✗	✗	✗
MAPO [15]	✗	✓	✓	✗	✗	✗	✗	✗	✗
PO2G [47]	✗	✓	✓	✗	✗	✗	✗	✗	✗
TRAS [17]	✗	✓	✓	✗	✗	✗	✗	✗	✗
StraGo [82]	✗	✓	✓	✗	◐	✗	✗	✗	✗
TextGrad [87]	✗	✓	◐	✗	✗	✗	◐	✗	✗
LLM-AutoDiff [85]	✓	✓	◐	✗	✗	✗	◐	✗	✗
GRAD-SUM [3]	✗	✓	◐	✗	✗	✗	✗	✗	✗
Trace [11]	✗	✓	✗	✗	✗	✗	✗	✗	✗
Evoke [35]	✗	✓	✗	✗	✗	✗	✗	✗	✗
ContraPrompt [65]	✗	✓	◐	✗	✗	✗	✗	✗	✗
MAPGD [29]	✗	✓	✗	✗	✗	✗	✗	✗	◐
PromptAgent [79]	✗	✓	✗	✗	✓	✗	✗	✗	✗
MPO [70]	✓	✓	✗	✓	✗	✗	✗	✗	✗
SAPO [42]	✓	✓	◐	✓	✓	✗	✗	✗	✗
UniPrompt [37]	✓	✓	✗	✓	✗	✗	✗	✗	✗
Compile-time / module-level									
DSPy [39]	✓	✓	✗	✗	✗	✗	✗	✗	✗
MIPROv2 [60]	✓	✓	✗	✗	✗	✗	✗	✗	✗
Population-based									
EvoPrompt [28]	✗	✓	✗	✗	✓	✓	✗	✗	✗
Promptbreeder [25]	✗	✓	✗	✗	✓	✓	✗	✗	✗
GAAPO [69]	✗	✓	✗	✗	✓	✓	✗	✗	✗
GEPA [1]	✓	✓	✗	✗	✓	✓	✗	✗	✗
+ Merge [1]	✓	✓	✗	✗	✓	✓	✗	✗	✗
Preference- and cost-aware									
BPO [12]	✗	✗	✗	✗	✗	✗	✗	✗	✗
$\text{CAPO}_{\text{cost}}$ [88]	✗	✓	✗	✗	✓	✗	✗	✓	◐
$\text{CAPO}_{\text{constr}}$ [20]	✗	✓	✗	✗	✓	✗	✗	✓	◐
SPLICE solver (this paper)	✓	✓	✓	✓	✓	✓	✓	✓	✓

parent can fill it. (On SST-5 under the protocol we reproduce, the published GEPA number, 42.0, is below the unoptimized seed's 44.6; we report it as published.)

Self-improvement loops. The idea of a system that rewrites its own components dates to the Gödel machine [68], which accepts a self-modification only when it is provably beneficial. LLMs have made empirical versions of this loop practical. STOP [90] lets a scaffolding program improve itself, and Gödel Agent [86] lets an agent modify its own logic. ADAS [34] has a meta-agent write new agents in code and keeps an archive of discoveries, the Darwin Gödel Machine [92] grows an archive of coding agents that edit their own source, and AFlow [91] searches agentic workflows with Monte Carlo tree search. FunSearch [66] and AlphaEvolve [59] pair an LLM proposer with an automatic evaluator to evolve programs. Promptbreeder [25] is the self-referential case within NLO, evolving the prompts that mutate prompts. These loops share the propose–score–retain skeleton of Definition 14, and most depart from it in the ways Section C names for GEPA and OPRO: proposals conditioned on an archive rather than one parent, and archives that grow without bound. Program search also anticipates selective mutability. FunSearch evolves a single function inside a fixed program skeleton, and AlphaEvolve evolves only the code blocks a user marks, leaving the rest fixed. What NLO has lacked is that design at the granularity of a section inside a prompt, together with an account of what freezing costs; self-modifying agent loops, where the modified system is also the modifier, otherwise enforce such limits outside the search.

Test-time search. A parallel line spends compute at inference time on each query rather than once on the artifact. Self-consistency [78] and repeated sampling [8] aggregate or verify many samples, Tree of Thoughts [84] searches over intermediate reasoning, and self-critique refines a single response [55, 72]. Snell et al. [74] show that allocating it adaptively between parallel samples and sequential revisions can be more effective than scaling model parameters. NLO moves the same search off the query path: the cost is paid once, and the improved artifact conditions every later query. The parallel-versus-sequential question reappears in our setting as the comparison between Best-of-N sampling and iterative search under a matched proposer (Section 4).

Structure and constraints. Recent work decomposes a prompt into sections: MPO [70] applies gradients to named sections, SAPO [42] edits role, context, task and format segments separately, and UniPrompt [37] edits task facets and compares greedy and beam variants, in effect a selector ablation in a structured space. None makes a segment unreachable or prices length. Cost-aware CAPO [88] penalizes relative token length in the objective, the same shape as our $F - \lambda\kappa$, and we add the bias bound of Theorem 5; BPO [12] rewrites prompts toward human preferences in a single pass. Constraint-aware CAPO [20], concurrent with this work, enforces tool-use, brevity and compliance requirements with a primal–dual method that *prices* violations, so a contract-breaking prompt remains feasible. Our mutability tags make it *unreachable* instead, and we reserve pricing for length, where growth is undesirable but not forbidden. To our knowledge no prior NLO method makes a section inside a prompt unreachable while the rest is optimized; program search has done the analogue for code, as discussed above.

B.2 Optimization Foundations

Combinatorial and noisy search. Greedy maximization of monotone submodular functions achieves $(1 - 1/e)$ [57], extended to approximately submodular objectives by the submodularity ratio [16] [see also 41, 56]; we use it for block-level greedy search and the price of freezing. Choosing among near-tied candidates on a finite batch is best-arm identification [2, 24, 38], with random search [5] as a baseline. Beam search [52], simulated annealing [40] and tabu search [26] are selectors over a shared state space and neighborhood [33], the abstraction our edit graph adopts. Local search on tours [30, 48, 49], neural architecture search [23, 50, 63, 95] and model-based hyperparameter optimization [6, 36] search structured spaces under expensive, noisy evaluation, and evolutionary computation supplies classical diversity mechanisms [18, 27, 31, 43]; our diversity slots act on parentage.

Theory of iterative search. Runtime analysis of evolutionary algorithms bounds the expected time for elitist search to reach an optimum [21], and our convergence result is in the same spirit, with a measurable proposer-competence condition in place of a fixed mutation operator. For prompt optimization, existing analyses bound which candidate to evaluate or keep under noise [45, 53], the convergence of continuous relaxations [29, 77], or the choice of evaluation data [58]. None analyzes the search itself: whether the retained candidate can regress, whether a population collapses onto one parent's children, or what a constraint costs. These are properties of a selector on a structured space, and Section E treats them.

Notes on Table 5. The rows cover every prompt optimizer we discuss or report a number for; program-search systems (FunSearch, AlphaEvolve) are omitted since they optimize code, as are the hand-written prompts in the benchmark tables (task description, expert and CoT prompts). "Structure": structured or modular prompt representation. "DualGrad": both positive and negative textual gradients. "SecLocal": gradients or updates at the level of named prompt sections. "Elitism": parents can survive a step. "Diversity": an explicit mechanism for preserving population diversity. "Immutable": designated sections cannot be modified by any reachable edit. "Priced": prompt length or cost enters the objective. "Theory": formal guarantees about the search; ◐ for ProTeGi marks a bandit selection procedure rather than a result about the search trajectory. ◐ under "Immutable" marks a whole-variable freeze (TextGrad's `requires_grad=False`, AdalFlow's `requires_opt=False`) rather than a section inside a prompt. GEPA+Merge adds a system-aware crossover step to GEPA and coincides with it on every axis tabulated here. Two unrelated methods are both named CAPO: $\text{CAPO}_{\text{cost}}$ [88] is cost-aware and prunes candidates under a token budget, while $\text{CAPO}_{\text{constr}}$ [20] is concurrent constraint-aware work. Both earn "Priced", and both retain candidates by truncation over parents and offspring, so both earn "Elitism"; neither can render a section unreachable, since

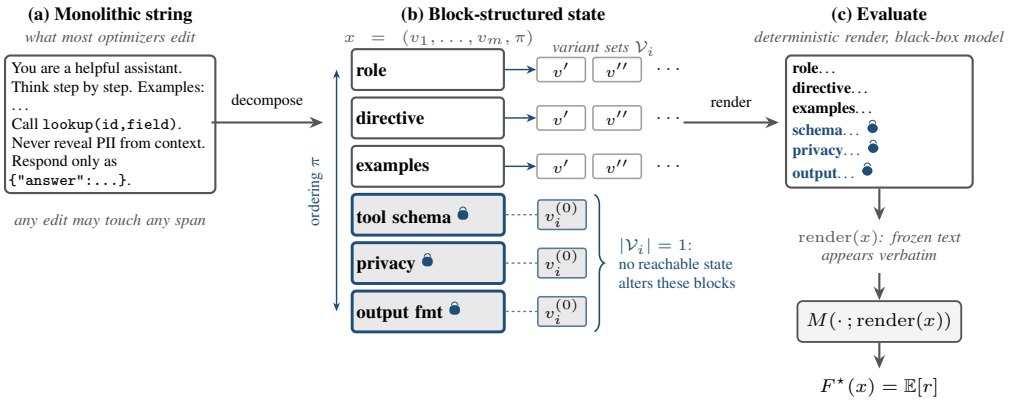


Figure 5: **A prompt as a block-structured state. (a)** Most prior optimizers edit a deployed prompt as one string, so any proposed rewrite may touch any span, including the tool schema that downstream parsers consume and the privacy clause that survived review. **(b)** We represent the same prompt as a state $x = (v_1, \ldots, v_m, \pi)$: one variant per semantically coherent block, plus an admissible ordering (Definition 5). Mutable blocks (white) range over variant sets $\mathcal{V}_i$ that the proposer explores. Immutable blocks (shaded, locked) satisfy $|\mathcal{V}_i| = 1$, so no state in the search space differs from the seed on those coordinates (Definition 3); the space itself preserves the contract *text*; whether the model still obeys it is measured in Section 4. **(c)** A deterministic render maps the state to a string, which the black-box model consumes to produce reward (Definitions 6 and 9). Because render is deterministic, frozen blocks appear in the rendered prompt verbatim. Most prior work optimizes the object in (a); the constrained problem of Definition 11 is posable only over the object in (b).

CAPO$_{\text{constr}}$ enforces its constraints as Lagrangian penalties. SAPO instructs its generator to preserve segments judged strong, which we do not count as "Immutable" because the protection is a request to the proposer rather than a property of the space (Section 2).

C Framework Details

This section supplements the framework of Section 2 with the two statements the main text cites but does not state, the edit-graph diameter (Proposition 3) and the well-definedness of proposals (Assumption 2), and with how existing methods sit in the batched search template as selectors (Section C.1).

Proposition 3 (Diameter of the edit graph)**.** *Let* $\mathrm{diam}(\Pi)$ *denote the diameter of the permutation subgraph induced by the chosen ordering moves. Then*

$$\mathrm{diam}(\mathcal{G}) \;\le\; \sum_{i=1}^{m} \mathbf{1}\{|\mathcal{V}_i| \ge 2\} \;+\; \mathrm{diam}(\Pi). \tag{10}$$

Assumption 2 (Well-defined proposal distribution)**.** *For every* $x \in \mathcal{X}$ *and history* H_t, $q_\phi(\cdot \mid x, H_t)$ *is a probability distribution supported on a subset of* $\mathcal{N}(x) \cup \{u_\varnothing\}$*: no proposal can leave* $\mathcal{X}$ *or write to an immutable block.*

C.1 Algorithms as Selectors

The selectors compared in Section 4 are instances of Definition 14 that differ in the retention rule. With $\Psi_t = \tilde{F}(\cdot\,; \mathcal{D}_{\text{valid}})$ and the final-leader output rule: *Best-of-*N is a single step (T=1, W=1, k=N) that keeps the best child and drops the seed; *greedy* is beam search with W=1, $\mathrm{top}_1(\mathcal{F} \setminus \mathcal{C}_{t-1}, \Psi)$: the best child replaces the parent even when it scores lower; *beam search* in the style of ProTeGi keeps the top-W *children*, $\mathrm{top}_W(\mathcal{F} \setminus \mathcal{C}_{t-1}, \Psi)$, dropping parents each step; *elitist beam search* keeps

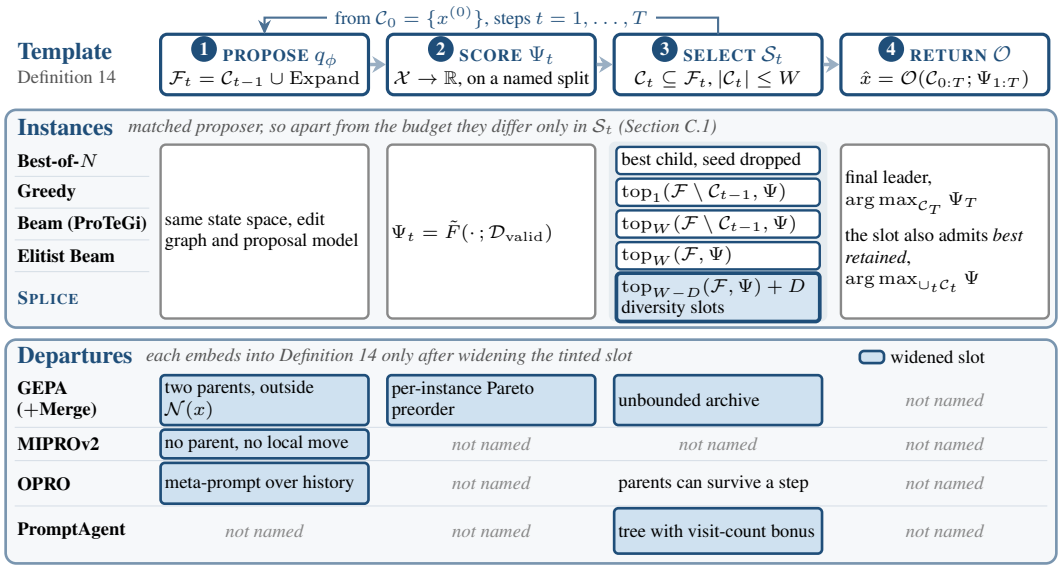


Figure 6: **One template, four slots, and what each method plugs in.** *Top:* batched iterative search (Definition 14). From $\mathcal{C}_0 = \{x^{(0)}\}$, each step expands the beam into a frontier with a proposal model q_ϕ, scores it with a scoring functional Ψ_t, keeps at most W candidates with a selector $\mathcal{S}_t$, and after T steps returns one artifact through an output rule $\mathcal{O}$. *Middle:* the selectors compared in Section 4 are instances that share every slot but the selector. *Bottom:* four published methods depart from the template, and the tinted cells are the slots each one widens. "Not named" marks a slot the paper does not specify for that method, not a claim that it matches the template.

the top-W of the *full* frontier, $\text{top}_W(\mathcal{F}, \Psi)$; and the SPLICE solver extends the elitist rule with a diversity reservation read from candidate annotations (Section 2).

The published baselines of Section 4 are not re-run inside the template; their numbers come from the protocols we reproduce. Several of them are also not instances of Definition 14, and it is more useful to name the difference than to stretch the template over it. GEPA scores candidates by a Pareto preorder over per-instance results and keeps an unbounded archive; GEPA+Merge additionally recombines two parents, a move outside $\mathcal{N}(x)$; MIPROv2 searches instruction–demonstration combinations without a parent or a local move; OPRO conditions each proposal on a meta-prompt over the top-scoring history rather than on one parent; and PromptAgent retains an unbounded tree with a visit-count bonus. OPRO is the closest case: it fits Definition 14 if the parent is taken as nominal and the meta-prompt is passed through H_t, which Definition 13 allows. Each method embeds into Definition 14 only after widening a named component (a vector-valued scoring functional, a multi-parent expansion, a parent-free proposal, an unbounded pool, a nominal parent), and we do not claim that the attribution argument above transfers to them.

Casting the methods this way exposes the axes along which they differ (beam size W, parent survival, diversity, scoring set and output rule), and Table 5 records several of them (elitism, diversity, structure) for each prior method. The axes internal to the SPLICE solver are ablated in Section 4; the external baselines are compared as whole systems.

D Solver Details

This section collects the pseudocode and illustrations of the SPLICE loop described in Section 2, and the details of proposal generation.

D.1 Proposals

The proposer is conditioned on the parent, its cached validation score, its critique, and the momentum context,

$$\big(\text{render}(x),\ \tilde{F}(x),\ \text{Crit}_t(x),\ \text{Mom}_t\big),$$

Algorithm 1 The SPLICE solver.

input : initial state $x^{(0)} \in \mathcal{X}$; proposal model q_ϕ; critique agent M_{crit}; beam width W; diversity slots D; quality-gate ratio $r \in [0,1]$; proposals per candidate k; horizon T; momentum window w; minimum improvement $\Delta_{\min}$; patience T_{pat}; training data $\mathcal{D}_{\text{train}}$; validation data $\mathcal{D}_{\text{valid}}$; regularizers $\{R_j\}$ with weights $\{\lambda_j\}$.

Initialize: $\mathcal{C}_0 \leftarrow \{x^{(0)}\}$; score $\tilde{F}(x^{(0)}) \leftarrow F(x^{(0)};\, \mathcal{D}_{\text{valid}})$.

Gradient history $G_0 \leftarrow \emptyset$.

for $t = 1, \ldots, T$ **do**

// i. Re-run the beam on a training batch (critique evidence only).

Sample $B_t \subseteq \mathcal{D}_{\text{train}}$; compute $F(x;\, B_t)$ and per-example outcomes $\forall x \in \mathcal{C}_{t-1}$.

// ii. Dual, section-local critique, one per parent.

$\text{Crit}_t(x) \leftarrow M_{\text{crit}}\big(x,\ B_t\text{-outcomes of } x\big) \quad \forall x \in \mathcal{C}_{t-1}$

// iii. Momentum context from the gradient trajectory and cached scores.

$\text{Mom}_t \leftarrow \textsc{BuildMomentum}\big(G_{t-1},\, \{\tilde{F}(\text{leader}(\mathcal{C}_s))\}_{s<t},\, \{\tilde{F}(x)\}_{x\in\mathcal{C}_{t-1}};\ w, \Delta_{\min}\big)$

// iv. Propose k candidates per beam member (edit only mutable blocks).

$\mathcal{N}_t \leftarrow \bigcup_{x\in\mathcal{C}_{t-1}} \big\{\text{Apply}(x, u_j) : u_j \overset{\text{i.i.d.}}{\sim} q_\phi(\cdot \mid x,\, \tilde{F}(x),\, \text{Crit}_t(x),\, \text{Mom}_t),\ j = 1, \ldots, k\big\}$

Remove from $\mathcal{N}_t$ any duplicate of a member of $\mathcal{C}_{t-1}$ or of an earlier element of $\mathcal{N}_t$.

// v. Evaluate new candidates on validation, once (cached scoring).

for $x \in \mathcal{N}_t$ **do**

$\tilde{F}(x) \leftarrow F(x;\, \mathcal{D}_{\text{valid}}) - \sum_j \lambda_j R_j(x)$

end

// vi. Elitist selection with diversity slots (Algorithm 2).

$\mathcal{C}_t \leftarrow \mathcal{S}_t^{\text{elit-div}}\big(\mathcal{C}_{t-1} \cup \mathcal{N}_t,\ \tilde{F};\ W, D, r\big)$

// vii. Record the leader change and the critique behind it.

$x_t^{\text{L}} \leftarrow \text{leader}(\mathcal{C}_t)$

$\Delta_t \leftarrow \tilde{F}(x_t^{\text{L}}) - \tilde{F}(\text{leader}(\mathcal{C}_{t-1}))$

$G_t \leftarrow G_{t-1} \cup \{(t,\, \mathbf{g}_t,\, \Delta_t)\}$, where $\mathbf{g}_t$ is the summary of $\text{Crit}_t(\text{parent}(x_t^{\text{L}}))$, or a no-change marker if $x_t^{\text{L}} \in \mathcal{C}_{t-1}$

// viii. Early stopping.

if *the leader has not gained more than* $\Delta_{\min}$ *over the last* T_{pat} *steps* **then**

break

end

end

return $\hat{x} \leftarrow \text{leader}(\mathcal{C}_t)$ at the last completed step

where Mom_t summarizes G_{t-1} together with recent leader scores and the current beam's scores. All of it is fixed before any step-t proposal is drawn, which is what makes a parent's k proposals i.i.d. given the history. The proposer sees validation *scores* but never a validation example, and samples k structurally valid edits per parent at non-zero temperature, each confined to the mutable blocks.

Section E gives the probability that a step's draws contain an improving one (Theorem 3) and what is known about terminal quality (Proposition 4 and Remark 4).

E Theory: Supporting Results and Proofs

This section extends Section 3: the notation, the assumptions the main text does not state, the results it only points to, and proofs of every result. Table 6 lists what each result rests on.

The analysis has two goals. First, to place the SPLICE solver on firm footing: each component of Section 2 other than the critique (selective mutability, elitist selection, diversity slots, and length regularization) is paired with a formal guarantee that makes its contribution precise rather than anecdotal. Second, to name what the experiments of Section 4 can check: some results predict a quantity an ablation measures, and others are not tested in this paper. All proofs are deferred to Section E.

Algorithm 2 $\mathcal{S}_t^{\text{elit-div}}$: elitist selection with diversity slots.

input : frontier $\mathcal{F} = \mathcal{C}_{t-1} \cup \mathcal{N}_t$; regularized score $\tilde{F}$; beam width W; diversity slots D; quality-gate ratio $r \in [0, 1]$; current step t.
output : next beam $\mathcal{C}_t \subseteq \mathcal{F}$, $|\mathcal{C}_t| \leq W$.

```
// Degenerate cases.
```
if $D \leq 0$ ***or*** $|\mathcal{F}| \leq W$ ***or*** $\mathcal{N}_t = \emptyset$ **then**
  return $\text{top}_W(\mathcal{F};\, \tilde{F})$
end
$D \leftarrow \min(D,\, W - 1)$; `// guarantee at least one elite slot`
```
// Elite slots by score.
```
$\mathcal{E} \leftarrow \text{top}_{W-D}(\mathcal{F};\, \tilde{F})$
$\ell \leftarrow \max_{x \in \mathcal{E}} \tilde{F}(x)$
$\tau \leftarrow r \cdot \ell$; `// quality floor relative to the leader`
$\Lambda \leftarrow \bigcup_{x \in \mathcal{E}} \big(\text{lineage}(x) \setminus \{\text{id}(x)\}\big)$
$P \leftarrow \{\text{parent}(x) : x \in \mathcal{E}\}$
$\mathcal{Q} \leftarrow (\mathcal{F} \setminus \mathcal{E})$ sorted by $\tilde{F}$ desc.
$\mathcal{K} \leftarrow \emptyset$
```
// Fill D diversity slots.
```
for $j = 1, \ldots, D$ **do**
  $c \leftarrow$ PICKDIVERSE($\mathcal{Q}$, Λ, P, t, τ) `// Algorithm 3`
  if c *is* none **then**
    break
  end
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{c\}$; $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{c\}$
  $\Lambda \leftarrow \Lambda \cup (\text{lineage}(c) \setminus \{\text{id}(c)\})$
  $P \leftarrow P \cup \{\text{parent}(c)\}$
end
```
// Backfill any unfilled diversity slots above the floor.
```
$\mathcal{K} \leftarrow \mathcal{K} \cup \text{top}_{D-|\mathcal{K}|}\big(\{c \in \mathcal{Q} : \tilde{F}(c) \geq \tau\};\, \tilde{F}\big)$
return $\mathcal{E} \cup \mathcal{K}$ sorted by $\tilde{F}$ desc.

Two optima, and which one every statement here targets. (5) defines $x^\star_{\text{all}}$ as the maximizer of $F^\star$ over the *all-mutable* space $\mathcal{X}_{\text{all}}$. That is the right reference for pricing the constraints (Proposition 2) and the wrong one for analyzing the search, which never leaves $\mathcal{X}$. Throughout this section, therefore,

$$x^\star \;\in\; \arg\max_{x \in \mathcal{X}} F^\star(x), \tag{11}$$

the optimum over the constrained space with contract blocks frozen (the $x^\star_{\text{mut}}$ of Proposition 2). The distinction is not cosmetic. Measured against $x^\star_{\text{all}}$, a state that is optimal within $\mathcal{X}$ may still be ε-suboptimal with no improving neighbor available, which would make Assumption 1 false and the target of Theorem 4 unreachable at any horizon; Section F.2 measures that residual gap at 10.5 pp on a deliberately expensive contract, so the difference is real rather than hypothetical. Every ε-optimality claim below is against the best *feasible* prompt, and the price of feasibility itself is Proposition 2's separate subject.

What the concentration results need from the metric. Relating a batch score to the population score takes two properties of the metric.

> **Assumption 3** (Bounded, decomposable metric)**.** *The per-example score lies in* $[0, 1]$ *(Definition 8), and the score of a batch is the mean of its per-example scores* (2)*.*

Boundedness means, by Hoeffding's lemma, that each centered score is sub-Gaussian with variance proxy $\sigma^2 \leq 1/4$ (Lemma 2). Decomposability is what lets a batch of n examples estimate the population score to within $O(n^{-1/2})$. Accuracy and exact match satisfy both. Set-level metrics, such as macro-F_1 and the mean Spearman ρ of the judge task (Section F.5), are not decomposable, because they are not averages of per-example terms; ρ also ranges over $[-1, 1]$ rather than $[0, 1]$.

Algorithm 3 PICKDIVERSE (`mode=lineage`): one diversity pick.

input : sorted-desc candidates $\mathcal{Q}$; used-ancestor set Λ; used-parent set P; current step t; quality floor τ.
output : a diverse candidate $c \in \mathcal{Q}$, or NONE.

```
// T1:  brand-new lineage, born this step.
for c ∈ Q do
    if F̃(c) < τ then
        break
    end
    if (lineage(c) \ {id(c)}) ∩ Λ = ∅ and gen(c) = t then
        return c
    end
end
// T2:  brand-new lineage, any generation.
for c ∈ Q do
    if F̃(c) < τ then
        break
    end
    if (lineage(c) \ {id(c)}) ∩ Λ = ∅ then
        return c
    end
end
// T3:  at least a different immediate parent.
for c ∈ Q do
    if F̃(c) < τ then
        break
    end
    if parent(c) ∉ P then
        return c
    end
end
return NONE
```

Only the concentration results use Assumption 3, since they are the results that relate a batch score to the population score (Theorems 2 and 7 and Corollaries 1 and 2), and for set-level metrics they do not apply as stated. The remaining results (monotonicity, diversity, reachability, convergence on the scored objective, and the prices of regularization and mutability) do not relate a batch score to the population score, so they do not use Assumption 3 and hold for any scoring function.

Cached scores. SPLICE scores a candidate on $\mathcal{D}_{\text{valid}}$ when it is first evaluated and caches that score while the candidate stays in the beam; a state that leaves the beam and is later re-proposed verbatim is evaluated again (Section D). We call this the *cached-scoring convention*, and Lemma 1 needs nothing more. The definitions and results that treat $\tilde{F}(\cdot) := \tilde{F}(\cdot\,; \mathcal{D}_{\text{valid}})$ as a function on $\mathcal{X}$ (Definitions 15 and 17, Assumption 5, Theorems 3 and 4, Propositions 1 and 5, and Corollary 2) assume in addition that scores are *cached per state*: every evaluation of a state returns the same score, as with a deterministic system agent. Theorem 7(ii) covers the general case, where re-evaluations are priced by a factor $T + 1$. The exception is the re-evaluation regime of Theorem 2 and Corollary 1, the alternative design in which every frontier member is re-scored on fresh examples at each step.

E.1 Probabilistic Preliminaries

We record the standard facts used below, so that every proof reduces to an explicit application of one of them.

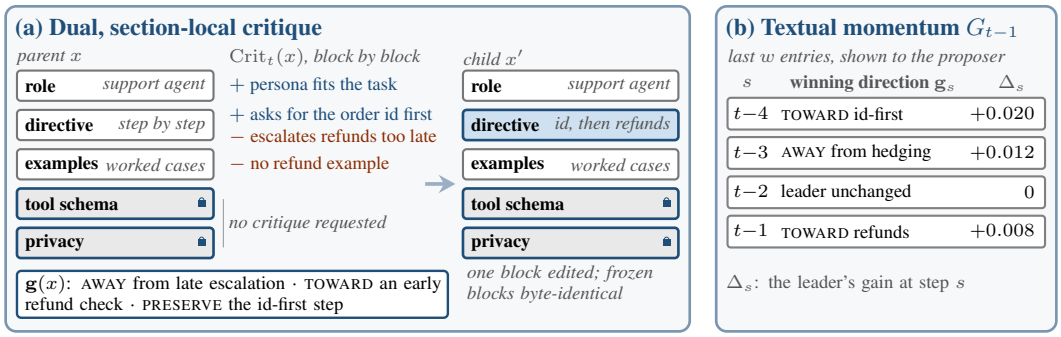


Figure 7: **The critique and the momentum context** (illustrative content, on the running example of Figure 1). **(a)** The critique agent reads a parent's traces on the training batch and returns, for each mutable block, what to keep ($\mathbf{p}_i$) and what to repair ($\mathbf{n}_i$), plus a directional summary $\mathbf{g}(x)$ (Equation (7)). Frozen blocks receive no critique. The proposer edits the block the critique points at, here the directive, and leaves the frozen blocks byte-identical. **(b)** The proposer also sees the momentum context, the last w entries of the gradient trajectory (Equation (8)): the direction behind each new leader and the leader's gain Δ_s, so it can keep following a direction that worked.

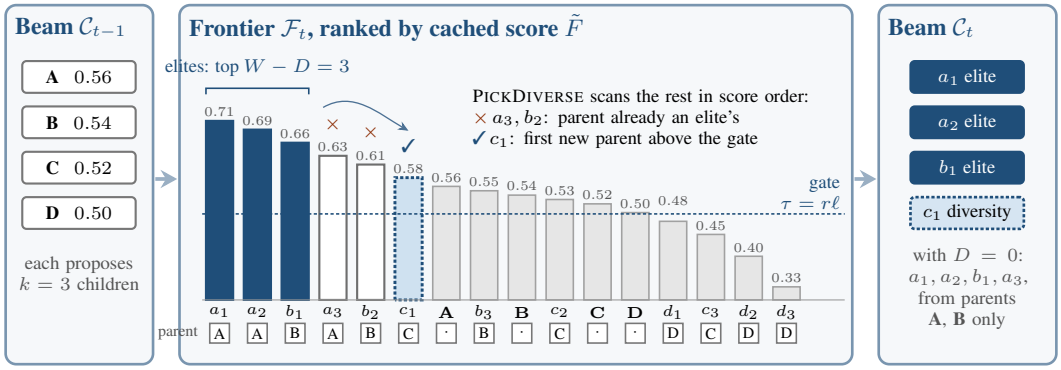


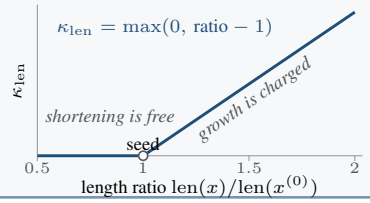
Figure 8: **One selection step, with numbers** ($W = 4$, $D = 1$, $k = 3$, $r = 0.7$). The four beam members each propose three children, and parents and children are ranked together on scores cached at first evaluation (Algorithm 2). The top $W - D = 3$ fill the elite slots; all are children of **A** or **B**. The leader score is $\ell = 0.71$, so the quality gate is $\tau = r\ell \approx 0.50$ (dashed line). Carried beam members (**A**–**D**) have parents from an earlier step, shown as ·. PICKDIVERSE (Algorithm 3) scans the remaining candidates in score order and skips a_3 and b_2, whose parents are already elites' parents; c_1 is the first candidate above the gate with a new parent and takes the diversity slot. This is Tier 3: Tiers 1–2 admit nothing here, because the seed is not in this frontier and every other candidate descends from it. Had no candidate above τ had a new parent, the slot would be backfilled with the best remaining candidate above τ. With $D = 0$ the next beam would come from two parents only.

Lemma 2 (Hoeffding's inequality)**.** *Let $Z_1, \ldots, Z_n$ be independent with $Z_j \in [0,1]$ and $\bar{Z} = \frac{1}{n}\sum_j Z_j$. Then for any $\varepsilon > 0$, $\Pr\big[|\bar{Z} - \mathbb{E}\bar{Z}| > \varepsilon\big] \le 2\exp(-2n\varepsilon^2)$. Moreover $\bar{Z} - \mathbb{E}\bar{Z}$ is sub-Gaussian with variance proxy σ^2/n where $\sigma^2 \le 1/4$.*

Lemma 3 (Difference of independent sub-Gaussians)**.** *If U, V are independent, mean-zero, sub-Gaussian with variance proxies s^2 each, then $U - V$ is mean-zero sub-Gaussian with proxy $2s^2$, so $\Pr[U - V > \varepsilon] \le \exp\big(-\varepsilon^2/(4s^2)\big)$.*

Proof. The moment generating function of a sub-Gaussian with proxy s^2 satisfies $\mathbb{E}e^{\theta U} \le e^{\theta^2 s^2/2}$. By independence $\mathbb{E}e^{\theta(U-V)} \le e^{\theta^2 s^2/2} e^{\theta^2 s^2/2} = e^{\theta^2(2s^2)/2}$, which is the proxy-$2s^2$ bound; the tail bound is the Chernoff optimization $\theta = \varepsilon/(2s^2)$. □

(a) The cost



(b) What λ changes

candidate	$F(x; \mathcal{D}_{\text{valid}})$	ratio	$\tilde{F}$, $\lambda = 0$	$\tilde{F}$, $\lambda = 0.05$
parent x	0.80	1.00	0.800	**0.800**
child x'	0.81	1.30	**0.810**	0.795

The child gains 0.01 but is 30% longer. At $\lambda = 0.05$ the growth costs $0.05 \times 0.30 = 0.015$, more than the gain, so the selector keeps the parent. Whatever λ is, the accuracy given up is at most $\lambda \cdot \kappa(x^\star)$ (Theorem 5).

Figure 9: **Pricing growth** (illustrative numbers). **(a)** The ratio cost of (4) is zero at or below the seed's length and grows linearly beyond it, so it has the same meaning on every task. **(b)** A child that buys a small gain with a long addition wins at $\lambda = 0$ and loses at $\lambda = 0.05$; the tinted cells mark the candidate the selector prefers at each λ.

Table 6: **Which results depend on proposal coverage, and which do not.** Proposal coverage, in its population form (Assumption 1) and its scored form (Assumption 5), is the assumption about the LLM proposer that we cannot verify. It is used by three results, all of which bound a *rate* (bottom block); the top block holds whether or not the proposer is competent (Section E). The middle block rests on γ-submodularity (Assumption 4) instead, a different and in one respect weaker position, since we have no per-benchmark estimator for γ (Section 5).

Result	Rests on	Coverage?
Coverage-free: hold whether or not the proposer is competent		
Lemma 1 (leader well-definedness)	cached scoring, parent retention	✗
Proposition 3 (diameter)	connected ordering moves	✗
Proposition 5 (beam vs greedy)	coupled draws, per-state cache	✗
Theorem 1 (diversity)	a qualifying candidate in $\mathcal{F}_t$	✗
Theorem 2 (noisy no-regression)	Assumption 3, fresh batches, elitist selector	✗
Corollary 1 (minimum batch)	as above	✗
Theorem 7 (uniform convergence)	Assumption 3; (ii) single-edit proposals	✗
Theorem 5 (regularization bias)	$\kappa \geq 0$ only	✗
Proposition 1 (adaptive annealing)	Lemma 1, per-state cache	✗
γ-submodularity (Assumption 4), not coverage		
Proposition 4 (greedy exactness)	γ-submod. + one alternative per block	✗
Proposition 2 (price of freezing)	γ-submod. on E^{all} + diminishing returns	✗
*Coverage-dependent: the **rate** results only*		
Theorem 3 (single-step reachability)	Assumptions 1 and 2, per-state cache	✓
Theorem 4 (convergence rate)	Assumption 5 (gain $\Delta_{\min}$), Lemma 1, per-state cache	✓
Corollary 2 (population form)	as above + Theorem 7	✓

Lemma 4 (Lower-tail Chernoff bound, explicit form)**.** *Let N_T be a sum of T Bernoulli indicators, each with conditional success probability at least $p \in (0, 1]$ given the past. Fix an integer $K \geq 1$ and $\delta \in (0, 1)$. If*

$$T \;\geq\; \frac{2K + 8\ln(1/\delta)}{p}, \tag{12}$$

then $\Pr[N_T < K] \leq \delta$.

Proof. Because each indicator has conditional success probability at least p given the past, N_T stochastically dominates $\text{Bin}(T, p)$: couple each indicator to a $\text{Bernoulli}(p)$ variable that it dominates pointwise, and sum. It therefore suffices to bound the lower tail of $Y \sim \text{Bin}(T, p)$ with mean $\mu = Tp$. The multiplicative Chernoff bound gives, for $0 < K < \mu$,

$$\Pr[Y < K] \;\leq\; \exp\!\left(-\tfrac{(\mu - K)^2}{2\mu}\right).$$

Write $\beta = \ln(1/\delta)$. By (12), $\mu \geq 2K + 8\beta$. Then $K \leq \mu/2$, so $\mu - K \geq \mu/2$ and $\frac{(\mu-K)^2}{2\mu} \geq \frac{(\mu/2)^2}{2\mu} = \frac{\mu}{8} \geq \frac{2K+8\beta}{8} \geq \beta$. Hence $\Pr[Y < K] \leq e^{-\beta} = \delta$. □

Table 7: **The results of Section E at a glance.** Each row gives a result, what it says, and what it rests on. Proposal coverage, in its population form (Asm. 1) and its scored form (Asm. 5), is the assumption about the LLM proposer that we cannot verify; only the three rate results in the bottom block use it. The middle block rests on γ-submodularity (Asm. 4) instead, for which we have no per-benchmark estimator.

Result	What it says	Rests on	Cov.
Hold whether or not the proposer is competent			
Prop. 3	*Diameter.* Any two states are at most $m_{\mathrm{mut}} + \mathrm{diam}(\Pi)$ edits apart	connected ordering moves	✗
Lem. 1	*Leader.* The leader's score never decreases	cached scoring, parent retention	✗
Prop. 5	*Beam vs greedy.* The elitist leader scores at least greedy's state until the beam drops that state	coupled draws, per-state cache	✗
Thm. 1	*Diversity.* If a competitive child of another parent exists, the beam keeps one	a qualifying candidate in $\mathcal{F}_t$	✗
Prop. 1	*Gate.* The quality gate $\tau_t = r\ell_t$ never falls	Lem. 1, per-state cache	✗
Thm. 2	*Noisy no-regression.* Under re-scoring, an ε-regression has probability at most $W(1+k)e^{-n\varepsilon^2/c}$	Asm. 3, fresh batches, elitist selector	✗
Cor. 1	*Batch size.* Batch size $n \geq (c/\varepsilon^2)\log(W(1+k)/\delta)$ suffices for δ-monotonicity	as Thm. 2	✗
Thm. 7	*Uniform convergence.* Validation scores over the search history are within ε of population scores	Asm. 3; (ii) single-edit proposals	✗
Thm. 5	*Length bias.* Regularizing with λ costs at most $\lambda \cdot \kappa(x^\star)$ in utility	$\kappa \geq 0$	✗
Rest on γ-submodularity, not coverage			
Prop. 4	*Greedy exactness.* With one alternative per block, block-pass greedy reaches the optimum	$\gamma > 0$, one alternative per block	✗
Prop. 2	*Price of freezing.* Freezing costs at most γ^{-1} times the frozen blocks' best single-block gains	γ on E^{all}, diminishing returns	✗
Conj. 1	*Beam width.* At each γ, worst-case leader quality is non-decreasing in W	γ-submodular instances	✗
Rate results: need proposal coverage			
Thm. 3	*Reachability.* A step finds an improving child with probability at least $1-(1-p^\star)^{\lvert\mathcal{C}_{t-1}\rvert k}$	Asm. 1, 2, per-state cache	✓
Thm. 4	*Convergence.* Horizon $T \geq (2K + 8\ln(1/\delta))/p^{\mathrm{adv}}$ steps reach ε-optimality with probability $1-\delta$	Asm. 5, Lem. 1, per-state cache	✓
Cor. 2	*Population form.* The same, stated for population utility	as Thm. 4, plus Thm. 7	✓

Lemma 4 gives Theorem 4 an explicit requirement, (12), in place of an unspecified universal constant; it is additive rather than multiplicative in $\ln(1/\delta)$.

E.2 Edit-Graph Reachability

The diameter bound Proposition 3 is a worst-case statement about the geometry of $\mathcal{X}$: any two states are connected by at most $m_{\mathrm{mut}} + \mathrm{diam}(\Pi)$ primitive edits, and selective mutability reduces this by excluding immutable blocks from the replacement phase. But SPLICE does not enumerate neighborhoods; it samples from q_ϕ, and progress at step t depends on whether an *improving* neighbor is among the Wk samples drawn.

Definition 15 (Improving proposal)**.** *For a state x and history H_t, a proposal $u \sim q_\phi(\cdot \mid x, H_t)$ is* improving from x *if* $\tilde{F}(\mathrm{Apply}(x,u)) > \tilde{F}(x)$.

Assumption 1 is stated in the main text (p. 7).

Assumption 1 is a *minimum-competence* condition: whenever a sufficient improvement is available, one is proposed with non-vanishing probability. It is not a property we prove about the LLM. Three

clarifications matter, because the form of the statement carries most of what a reader should take from it.

It is required at one tolerance, and only away from the target. Asking for a single $p^\star$ bounded away from zero at *every* non-optimal state would not be plausible: near the optimum the remaining gap is small, most single-coordinate edits are neutral or worse, and the improving probability decays toward zero. The assumption avoids that only because ε is fixed in advance. Were it required for every $\varepsilon > 0$, it would reinstate the implausible version: any non-optimal state lies in the sublevel set at ε equal to its own gap. With ε fixed, coverage is invoked only on states at least ε from the target, which is all Theorem 3 uses, since it is stated for that ε; the convergence rate rests on the scored analogue, Assumption 5, in the same form. The decay then shows up as a comparison *across* instantiations: a practitioner who demands a closer solution assumes coverage at a smaller ε, which is a weaker landscape and a larger step requirement in Theorem 4. That is the graceful degradation familiar from stochastic local search, and Section F.4.1 shows the improving rate falling as the leader climbs.

It constrains the landscape, the sampler and the scoring noise jointly. Three things must hold for $p^\star(\varepsilon) > 0$. The state must have an improving single-coordinate neighbor; at a radius-one local maximum that is still ε-suboptimal it does not, and the block-interaction phenomenon noted above is exactly the mechanism producing such states. The proposer must put positive probability on such a neighbor, which Assumption 2 does not give, since it bounds the support of q_ϕ from above rather than below. And because improvement is judged on the scored objective, the neighbor's validation score must exceed its parent's, which depends on the realized validation sample as well as on the population gap. The sublevel set is defined by population utility because the reachability argument reasons about which parents are truly suboptimal; the event inside is judged on the score because that is what the selector sees.

Coverage is needed at every parent, not just one. The reachability bound multiplies failure probabilities across all $|\mathcal{C}_{t-1}|\,k$ draws, and each factor is at most $1 - p^\star(\varepsilon)$ only if the parent it was drawn from is itself ε-suboptimal. That is why Lemma 5 asks it of every parent. If some parent is already ε-optimal the hypothesis fails and the theorem says nothing, though the beam then holds an ε-optimal candidate, which may not be the leader. Section F.4.1 estimates the improving rate directly from the per-step proposal acceptance rate.

Which results depend on coverage. Coverage, in its population form (Assumption 1) and its scored form (Assumption 5, stated with the convergence rate), is the assumption in this section we cannot verify, so it matters exactly where it is used: the population form in reachability and the scored form in the convergence results, all of which bound a *rate*. Table 7 makes the dependency structure explicit. The main consequence is that everything a practitioner relies on for safety is coverage-free. The returned prompt never scores below the seed on the validation split, the beam keeps a competitive distinct-parent candidate whenever one is proposed, and the text of frozen blocks is never modified, whether or not the proposer is competent. When coverage fails, SPLICE makes no progress and returns its monotone best-so-far rather than becoming unsafe. *Coverage failure costs speed, not safety.*

Coverage is also not a convenience specific to SPLICE. It is a condition on the problem class: if no improving edit can be proposed from a reachable non-optimal state, no proposer–critic loop can progress from it, so the limitation belongs to the paradigm rather than to our analysis of it. Prior work does carry formal guarantees: ProTeGi's candidate selection is a bandit procedure, $\text{CAPO}_{\text{cost}}$'s survival step uses racing with a multiple-testing correction, and constrained variants prove compliance results under their own assumptions. These guarantees concern selecting among candidates, not the trajectory of the search that produced them, so they do not need a condition on what the proposer can generate. We name that condition and measure it.

Theorem 3 is stated in the main text (p. 7).

The bound is on the draws, before deduplication. An improving draw whose child duplicates a member of $\mathcal{C}_{t-1}$, or an earlier draw of the step, is removed from $\mathcal{N}_t$; the state it reached is then already in the beam or in $\mathcal{N}_t$, so it adds no new candidate.

The exponent counts the draws made, $|\mathcal{C}_{t-1}|\,k$, and not the budget Wk; the two coincide only once the beam is full. They do coincide for the whole run after the first step whenever $1 + k \geq W$, as at $W{=}4$ with $k{=}3$ (GPQA and the seeded SST-5 sweeps) or $k{=}5$ (BBH), since the step-1 frontier

holds $1 + k \geq W$ candidates and elitist selection retains W of them, so $|\mathcal{C}_{t-1}| = W$ for every $t \geq 2$ whenever the frontier supplies W distinct candidates. At $t = 1$ the exponent is k, because $\mathcal{C}_0 = \{x^{(0)}\}$ is a single parent. We state the theorem in the exact form rather than the budgeted one because substituting Wk at $t = 1$ would overstate the first step's guarantee, and the first step is where the run has the least to fall back on.

Theorem 3 quantifies the joint effect of beam width and proposals-per-candidate: increasing either drives the failure probability to zero exponentially. It is a statement about *some* parent improving, and it is the wrong quantity for predicting whether the search makes progress. At $Wk = 12$ a per-proposal rate of 0.25 gives $p^{\text{step}} = 0.97$ and even 0.08 gives 0.63, so by this measure fan-out looks almost freely substitutable for proposer quality. Progress of the *leader*, which is what Theorem 4 counts, runs on the leader's own k draws, and there the same rates give 0.58 and 0.22 at $k = 3$. So the operating condition is $p^\star(\varepsilon)$ above $1/k$, not $1/Wk$, and beam width buys reachability without buying the proven rate. The coverage rate is also not a property of the proposer alone: Section F.4.1 measures it with the proposer fixed and finds it rises with the capacity of the target model, so where a given proposer falls relative to that condition depends on what it is prompting.

Proofs.

Proof of Proposition 3. Fix $x = (v, \pi)$ and $x' = (v', \pi')$ in $\mathcal{X}$, where v, v' are the block-variant assignments and π, π' the orderings. We exhibit a path in the edit graph of the stated length.

A block-replacement move alters one coordinate v_i and leaves π fixed; an ordering move alters π and leaves every v_i fixed. The two families therefore act on disjoint coordinates of the pair (v, π) and commute as operators on $\mathcal{X}$, so we may schedule all replacements before all ordering moves without changing the endpoint.

Replacement phase. For each i with $v_i \neq v'_i$, one block-replacement move sends $v_i \mapsto v'_i$; this is admissible because $v'_i \in \mathcal{V}_i$ by $x' \in \mathcal{X}$. Immutable blocks satisfy $v_i = v_i^{(0)} = v'_i$ by Definition 3 and so contribute no move. A block with $|\mathcal{V}_i| = 1$ likewise cannot differ. Hence the number of replacement moves is at most $\sum_{i \in I_{\text{mut}}} \mathbf{1}\{|\mathcal{V}_i| \geq 2\} \leq m_{\text{mut}}$.

Ordering phase. The reachable orderings form the connected set Π under the admissible ordering moves, so π and π' are joined by at most $\operatorname{diam}(\Pi)$ such moves.

Concatenating the two phases gives a path of length at most $m_{\text{mut}} + \operatorname{diam}(\Pi)$, and since x, x' were arbitrary this bounds the diameter, which is (10). Selective mutability enters only through the replacement phase: freezing block i removes it from I_{mut} and hence deletes one unit from the bound. □

The exponent in Theorem 3 counts every draw made in the expansion, $|\mathcal{C}_{t-1}|\, k$, and each draw only carries the coverage guarantee if the parent it was drawn from is itself ε-suboptimal. We isolate that condition first.

Lemma 5 (When every parent carries the coverage guarantee)**.** *Suppose* $\max_{x \in \mathcal{C}_{t-1}} F^\star(x) \leq F^\star(x^\star) - \varepsilon$*, i.e. the beam contains no* ε*-optimal candidate. Then every* $x \in \mathcal{C}_{t-1}$ *lies in the* ε*-sublevel set of Assumption 1, and hence every draw* $u \sim q_\phi(\cdot \mid x, H_t)$ *from every parent is improving from its parent with probability at least* $p^\star(\varepsilon)$*.*

Proof. Immediate: if the maximum population utility over the beam is at most $F^\star(x^\star) - \varepsilon$ then each individual member satisfies $F^\star(x^\star) - F^\star(x) \geq \varepsilon$, which is precisely the constraint indexing the minimum in Assumption 1. □

Remark 1. *The hypothesis cannot be weakened to "the beam* leader *is* ε*-suboptimal". The leader is the maximizer of the* empirical *score* $\tilde{F}(\cdot; \mathcal{D}_{\text{valid}})$*, not of* $F^\star$*, so under scoring noise a beam whose leader is* ε*-suboptimal may still contain an* ε*-optimal member. Lemma 5 is stated in population terms for exactly this reason. If the hypothesis fails, the beam already contains an* ε*-optimal candidate, but that does not meet the conclusion of Theorem 4, which concerns the leader's score: the* ε*-optimal member need not be the leader.*

Proof of Theorem 3. Condition on $\mathcal{C}_{t-1}$ and the history H_t, and assume the hypothesis of Lemma 5. Index the draws of the expansion by (x, j) with $x \in \mathcal{C}_{t-1}$ and $j \in [k]$, so there are $|\mathcal{C}_{t-1}| \cdot k \le Wk$ of them.

Independence. Within a parent, the k draws are i.i.d. from $q_\phi(\cdot \mid x, H_t)$ by (6). Across parents, each parent's expansion is a separate invocation of the proposer with its own randomness, and all of them condition on the same fixed $(\mathcal{C}_{t-1}, H_t)$; hence the full collection $\{u_{x,j}\}$ is mutually independent given $(\mathcal{C}_{t-1}, H_t)$.

Failure probability. Let $A_{x,j}$ be the event that $u_{x,j}$ is improving from x in the sense of Definition 15. By Lemma 5, $\Pr[A_{x,j}] \ge p^\star(\varepsilon)$ for every (x, j). The event that the expansion contains no improving child is $\bigcap_{x,j} A^c_{x,j}$, and by independence

$$\Pr\Big[\bigcap_{x,j} A^c_{x,j}\Big] = \prod_{x,j} \Pr[A^c_{x,j}] \le \big(1 - p^\star(\varepsilon)\big)^{|\mathcal{C}_{t-1}|k}.$$

Taking complements gives Theorem 3. Feasibility of every draw (no proposal leaves $\mathcal{X}$ or writes to an immutable block) is Assumption 2, so every improving draw reaches a state of $\mathcal{X}$. The event is on the draws $\{u_{x,j}\}$, not on $\mathcal{N}_t$ after deduplication: a draw whose child duplicates a member of $\mathcal{C}_{t-1}$ or an earlier draw is removed, and the state it reached is already in the beam or in $\mathcal{N}_t$.

One tempting step is unavailable, although the budgeted form reads like a harmless simplification. Bounding $(1 - p^\star)^{|\mathcal{C}_{t-1}|k}$ above by $(1 - p^\star)^{Wk}$ does not follow from $|\mathcal{C}_{t-1}| \le W$: because $0 \le 1 - p^\star \le 1$, raising it to the *larger* exponent makes it *smaller*, so $(1 - p^\star)^{Wk} \le (1 - p^\star)^{|\mathcal{C}_{t-1}|k}$ and the substitution runs the wrong way. The budgeted form is therefore a valid bound only when the beam is full. It fails at $t = 1$, where $|\mathcal{C}_0| = 1$, and the exponent is k. □

Remark 2 (What reachability does and does not give)**.** *Theorem 3 bounds the probability that* some *child beats* its own parent*. That is weaker than the leader advancing: a child improving on a non-leader parent need not exceed ℓ_{t-1}. The distinction is what separates p^{step} from the leader-advance rate p^{adv} of Theorem 4.*

E.3 Approximate Submodularity and Greedy Approximation

Theorem 3 bounds the probability that some proposal improves on its parent, which need not move the leader, but is silent about *how much* progress is possible overall. A terminal-quality bound needs structural information about $F^\star$. We import the standard framework: approximate submodularity of the utility as a set function over block-variant commits. Fixing $\pi = \pi^{(0)}$, let

$$E \;=\; \big\{(i, v) : i \in I_{\mathrm{mut}},\; v \in \mathcal{V}_i \setminus \{v_i^{(0)}\}\big\}, \qquad E^{\mathrm{all}} \;=\; \big\{(i, v) : i \in [m],\; v \in \tilde{\mathcal{V}}_i \setminus \{v_i^{(0)}\}\big\}, \tag{13}$$

where $\tilde{\mathcal{V}}_i$ denotes the *counterfactual variant set* of block i (the variants block i would admit had it not been frozen), with $\tilde{\mathcal{V}}_i = \mathcal{V}_i$ for every $i \in I_{\mathrm{mut}}$. So E collects the alternative variants across mutable blocks, each tagged with its block so that equal text in two blocks gives two elements, and $E^{\mathrm{all}} \supseteq E$ additionally carries the alternatives that freezing removed. Call $A \subseteq E^{\mathrm{all}}$ *feasible* if it has at most one entry per block, and for feasible A let x_A substitute each $(i, v) \in A$ into $x^{(0)}$. The gain function, defined on feasible sets, is

$$g(A) \;:=\; F^\star(x_A) - F^\star(x^{(0)}), \qquad g(\emptyset) = 0, \tag{14}$$

well defined on every feasible subset of E^{all} because (2) depends on a state only through $\mathrm{render}(x)$ and so extends verbatim from $\mathcal{X}$ to the all-mutable space $\mathcal{X}_{\mathrm{all}}$ of (5). Prompt optimization over mutable blocks is constrained maximization of g on E; the larger ground set enters only in Proposition 2, to express what the restriction to E costs.

Two points about $\tilde{\mathcal{V}}_i$, since it is the one object here not already supplied by Section 2. First, it is what makes the price of freezing expressible at all. By Definition 3 an immutable block has $\mathcal{V}_i = \{v_i^{(0)}\}$, so any quantity written as a maximum over $\mathcal{V}_i$ for $i \in I_{\mathrm{imm}}$ is identically zero, and a bound in that form could only ever assert that freezing is free. The counterfactual set is the smallest addition that lets the question be posed. Second, it is *not observable*: recovering $\tilde{\mathcal{V}}_i$ would mean running the

search with the contract released, which is precisely what deployment forbids. E^{all} therefore buys a meaningful statement, not a measurable one (Remark 14). Search itself never touches $E^{\mathrm{all}} \setminus E$: every algorithm in this paper optimizes g over E alone.

We note the shape of the constraint on E now, because it matters for what the approximation guarantee below covers: the feasible sets are those with at most one entry per block, which form a *partition matroid* with one part per mutable block and capacity one, not a cardinality constraint.

Definition 16 (Submodularity ratio, 16)**.** *The* submodularity ratio *of g on a ground set E' is*

$$\gamma(g; E') \;:=\; \min_{\substack{L,S \subseteq E',\ L \cap S = \emptyset \\ L \cup S \text{ feasible},\ g(L \cup S) > g(L)}} \frac{\sum_{e \in S} [g(L \cup \{e\}) - g(L)]}{g(L \cup S) - g(L)} \;\in\; [0, 1], \tag{15}$$

the minimum running over the pairs on which g is defined. It is the ratio of Das and Kempe [16] restricted to the partition matroid; when every subset is feasible the two coincide. $\gamma = 1$ recovers standard submodularity; $\gamma = 0$ indicates fully complementary interactions.

Assumption 4 (Monotone γ-submodular utility)**.** *The gain function g of* (14) *is monotone non-decreasing on the feasible subsets of E^{all} ($g(A) \le g(A')$ whenever $A \subseteq A'$ and A' is feasible) and has submodularity ratio $\gamma(g; E^{\mathrm{all}}) \ge \gamma$ for some $\gamma \in (0, 1]$.*

We state Assumption 4 on the larger ground set because Proposition 2 compares across the frozen/unfrozen boundary and so needs γ to control marginals on frozen coordinates too. Nothing is lost for the results that live on E: since $E \subseteq E^{\mathrm{all}}$, the minimum in (15) runs over at least as many pairs on the larger set, so $\gamma(g; E^{\mathrm{all}}) \le \gamma(g; E)$ and the assumption implies $\gamma(g; E) \ge \gamma$. Proposition 4 and Conjecture 1 may therefore be read with γ on E. The cost of the stronger form is that γ becomes a property of a landscape that includes unobservable coordinates. We report no estimate of γ on either ground set: estimating it per benchmark would take block-subset ablations we have not run, and it remains open (Section 5).

Beyond its scope, Assumption 4 is a structural assumption on the task, not the algorithm: committing better variants helps, and block interactions are bounded.

The shape of the constraint decides what is provable, and the two regimes it admits behave very differently. When each mutable block has a single alternative to its seed text, every subset of E is feasible and the partition constraint does no work. This regime is an idealization: by Definition 2, $\mathcal{V}_i$ holds every variant the proposer could produce, so no run of SPLICE is in it. We analyze it because it isolates what $\gamma > 0$ buys, for an idealized greedy that sees the exact gain function. *Block-pass greedy* starts from $A_0 = \emptyset$; at each pass it evaluates $g(A_j \cup \{e\})$ for every remaining $e \in E$, commits the element with the largest positive marginal gain, and stops when no marginal is positive. In this regime the guarantee is not an approximation factor but an exact one.

Proposition 4 (Block-pass greedy is exact in the single-alternative regime)**.** *Suppose each mutable block contributes exactly one alternative, so that $E = \{(i, v_i) : i \in I_{\mathrm{mut}}\}$ and every subset of E is feasible. Under Assumption 4, block-pass greedy returns a commit set A_r with*

$$g(A_r) \;=\; \max_{A \subseteq E} g(A), \tag{16}$$

that is, x_{A_r} attains the best utility reachable by block replacement at the baseline ordering.

Two things follow, and the second is a limitation. First, in this regime a positive submodularity ratio is doing more work than an approximation guarantee would suggest: $\gamma > 0$ forbids exactly the configuration that traps greedy, a set of blocks that is valuable jointly while each of its members is worthless alone. Second, the regime is narrow. As soon as some block offers two or more alternatives the feasible sets form a non-trivial *partition matroid*, greedy must also choose *which* variant to commit per block, and no terminal-quality guarantee we are aware of covers deterministic block-pass greedy there: the best available result for monotone γ-weakly-submodular maximization under a matroid constraint is $(1 + 1/\gamma)^{-2}$ for a *randomized* greedy variant [10], and whether deterministic greedy attains any non-trivial ratio beyond cardinality constraints was open in that work. Remark 4

records what this does and does not justify. We therefore claim no approximation factor in the multi-alternative regime rather than import one that does not apply.

Wider beams maintain multiple partial configurations and can commit to coordinated multi-block edits greedy would reject. Since Proposition 4 leaves nothing to improve when a single alternative per block is on offer, the interesting claim lives in the multi-alternative regime:

Conjecture 1 (Beam improvement)**.** *In the multi-alternative regime, fix $\gamma \in (0, 1]$ and a proposer that puts positive probability on every feasible single-block edit, score with the exact gain g, and run elitist beam search of width W until no proposal can improve a beam member. Let $\alpha_\gamma(W)$ be the worst case, over instances whose submodularity ratio is at least γ, of the expected ratio between the returned leader's gain and $\max_A g(A)$ over feasible A. Then α_γ is non-decreasing in W, and, over the instances where some optimal commit set requires two or more blocks to move together, strictly increasing for $W \le m_{\mathrm{mut}}$.*

We state Conjecture 1 as a conjecture, and deliberately in qualitative form. A proof would require a coupling between the beam's W lineages that elitist selection makes dependent, so the product-of-failure-probabilities argument does not apply; and we do not conjecture a specific functional form for α_γ, because no deterministic-greedy baseline exists in this regime (Proposition 4) for such a form to interpolate from. What *is* provable is a pathwise comparison:

Proposition 5 (Elitist beam stays ahead of greedy until they separate)**.** *Run elitist beam search, with selector $\mathcal{S}^{\text{elit-div}}$ for any $D \le W - 1$ (which includes* SPLICE*), and greedy on the same proposal draws, with scores cached per state and shared by both runs: whenever greedy's state x_t^{g} lies in $\mathcal{C}_t$, both procedures see the same k children of it. Let $\nu := \min\{t : x_t^{\mathrm{g}} \notin \mathcal{C}_t\}$, with $\nu = \infty$ if no such step exists. Then*

$$\ell_t \;\ge\; \tilde{F}\big(x^{\mathrm{g}}_{\min(t,\nu)}\big) \qquad \text{for every } t \le T. \tag{17}$$

For $W = 1$ the two procedures coincide under a common tie-breaking rule, so $\nu = \infty$ and $\ell_t = \tilde{F}(x_t^{\mathrm{g}})$ for every t.

Here greedy is hill climbing with retention: it moves to its best child only if that child scores at least as high as its current state. The Greedy arm of Section 4 is the children-only beam with $W{=}1$ of Section C.1, which can regress, so Proposition 5 does not describe it.

The comparison stops at ν because greedy then proposes from a state the beam no longer holds, and the beam never scores those children; Remark 5 gives a three-step instance with $W = 2$ in which greedy finishes above the beam leader. The domination is also on the *scored* objective $\tilde{F}$ that both selectors act on, since a score ordering need not agree with the population ordering, and it needs the coupling: two runs with independent draws cannot be compared path by path.

Proposition 5 says that an elitist beam is never behind the greedy path while it holds that path, and is at least level with greedy at the step it drops it. It does not say the beam finishes ahead of greedy; Conjecture 1 is the empirically motivated claim that the benefit *scales* with W.

Proofs. The feasible sets of (14) are the subsets of $E = \{(i, v) : i \in I_{\mathrm{mut}},\, v \in \mathcal{V}_i \setminus \{v_i^{(0)}\}\}$ containing at most one element per block. These form a partition matroid with one part per mutable block and capacity one, which is *not* a cardinality constraint. The distinction collapses in one regime and bites in the other, and the two need separating before either can be argued. When every mutable block offers exactly one alternative, each part contributes one element, every subset is feasible, and the rank budget m_{mut} is non-binding; otherwise the matroid constraint is non-trivial. Proposition 4 covers the first case and Remark 4 states what is and is not known about the second.

Remark 3 (Why no reduction to a cardinality constraint is needed)**.** *One route to a terminal-quality bound fixes a designated alternative $\hat{v}_i$ per mutable block, observes that the resulting ground set $\hat{E} = \{(i, \hat{v}_i)\}$ makes the partition constraint vacuous and the cardinality budget $|\hat{E}| = m_{\mathrm{mut}}$ non-binding, and invokes Das and Kempe [16, Thm. 3] on the reduction. That route is available but pointless, because the same observation that permits it makes its conclusion*

strictly weaker than the truth. Once the feasible family is the full power set, the $(1-e^{-\gamma})$ factor is not tight: Proposition 4 shows greedy is exactly *optimal there, by a three-line argument that uses only $\gamma > 0$ and monotonicity and needs no external approximation theorem. The reduction also does not extend: it compares greedy against the best subset of $\hat{E}$ rather than against the optimum over E, and those differ as soon as a block offers a second alternative, which is the regime Remark 4 addresses.*

Proof of Proposition 4. By hypothesis $E = \{(i, v_i) : i \in I_{\mathrm{mut}}\}$ carries exactly one element per part, so every subset of E is feasible and the cardinality budget $m_{\mathrm{mut}} = |E|$ is non-binding. Let A_r be greedy's output. Greedy halts only when no remaining element has positive marginal gain, so $g(A_r \cup \{e\}) - g(A_r) \le 0$ for every $e \in E \setminus A_r$; by the monotonicity in Assumption 4 each such marginal is also non-negative, hence

$$g(A_r \cup \{e\}) - g(A_r) \;=\; 0 \qquad \text{for all } e \in E \setminus A_r. \tag{18}$$

Suppose for contradiction that $g(E) > g(A_r)$. Then $L := A_r$ and $S := E \setminus A_r$ are disjoint with $g(L \cup S) = g(E) > g(L)$, so (L, S) is admissible in the minimum defining $\gamma(g; E)$ in (15), and that definition gives

$$\gamma\,\big[g(E) - g(A_r)\big] \;\le\; \sum_{e \in E \setminus A_r} \big[g(A_r \cup \{e\}) - g(A_r)\big] \;=\; 0$$

by (18). Since $\gamma > 0$ this forces $g(E) \le g(A_r)$, contradicting the supposition. Hence $g(A_r) = g(E)$, and $g(E) \ge g(A)$ for every $A \subseteq E$ by monotonicity, so $g(A_r) = \max_{A \subseteq E} g(A)$; with $g(A) = F^\star(x_A) - F^\star(x^{(0)})$ from (14) this is (16).

The hypothesis $\gamma > 0$ is needed here for the conclusion to hold, not only to bound its loss: it rules out a set of blocks that is jointly valuable while each member is individually worthless, which is the only configuration that could strand greedy at a zero-marginal set. The conclusion fails at $\gamma = 0$. Take two blocks with $g(\emptyset) = g(\{a\}) = g(\{b\}) = 0$ and $g(\{a, b\}) = 1$: this is monotone with $\gamma(g; E) = 0$, greedy halts immediately and returns 0, and the optimum is 1. □

Remark 4 (The multi-alternative regime is a partition matroid, and is open)**.** *Proposition 4 assumes one alternative per mutable block, an idealization no run of* SPLICE *satisfies, since $\mathcal{V}_i$ holds every variant the proposer could produce (Definition 2). The assumption is a real restriction, and dropping it changes the problem class rather than the constant: when some $|\mathcal{V}_i| > 2$ the feasible sets form a partition matroid of rank m_{mut} that is no longer the full power set of the ground set, and block-pass greedy must additionally decide* which *variant to commit in each block, by an inner maximization that no cardinality-constrained surrogate records.*
We know of no terminal-quality guarantee covering deterministic greedy in that setting. The strongest available result for maximizing a monotone γ-weakly-submodular function under a matroid constraint is $(1 + 1/\gamma)^{-2}$, achieved by a randomized *greedy variant [10]; that work reports it as the first non-trivial guarantee beyond cardinality constraints and states that the deterministic greedy algorithm's ratio for such constraints was unknown. Two consequences for how this paper should be read. First, the relevant factor is* not *$\gamma/(1+\gamma)$, and the familiar $1/2$ for matroid-constrained* submodular *greedy does not transfer to the weakly submodular case by analogy: $(1 + 1/\gamma)^{-2}$ is $1/4$ at $\gamma = 1$. Second,* SPLICE*'s selector is deterministic, so even that result does not apply to it unmodified. We therefore state no approximation factor in the multi-alternative regime and treat the terminal quality of deterministic block-pass greedy under a partition matroid as open. Settling it, or showing that randomization is necessary, would strengthen Section E at the point it is currently weakest.*

Proof of Proposition 5. The claim is a statement about two algorithms run on shared randomness, so we first fix the coupling it is made under. At every step, the k draws greedy makes from its current state x_t^{g} are the same k draws the beam makes from x_t^{g} whenever $x_t^{\mathrm{g}} \in \mathcal{C}_t$. Scores are the cached validation scores $\tilde{F}(\cdot; \mathcal{D}_{\mathrm{valid}})$ of Section 2, cached per state and shared by the coupled runs, so a given state carries the same score in both. Since $\mathcal{C}_0 = \{x^{(0)}\} = \{x_0^{\mathrm{g}}\}$, $\nu \ge 1$.

Steps $t < \nu$. By definition of ν, $x_t^{\mathrm{g}} \in \mathcal{C}_t$, so $\ell_t = \max_{x \in \mathcal{C}_t} \tilde{F}(x) \geq \tilde{F}(x_t^{\mathrm{g}})$.

Step $t = \nu < \infty$. Since $x_{\nu-1}^{\mathrm{g}} \in \mathcal{C}_{\nu-1}$, the coupling puts greedy's k children of $x_{\nu-1}^{\mathrm{g}}$ into the beam's expansion, or into $\mathcal{C}_{\nu-1}$ if deduplication removed them as duplicates of beam members. Greedy's next state is either one of these children or $x_{\nu-1}^{\mathrm{g}}$ itself, so in every case $x_\nu^{\mathrm{g}} \in \mathcal{F}_\nu = \mathcal{C}_{\nu-1} \cup \mathcal{N}_\nu$. Elitist selection retains the top $W - D \geq 1$ members of $\mathcal{F}_\nu$ by score, and x_ν^{g} is not retained, so every retained elite scores at least $\tilde{F}(x_\nu^{\mathrm{g}})$; in particular $\ell_\nu \geq \tilde{F}(x_\nu^{\mathrm{g}})$.

Steps $t > \nu$. By Lemma 1, $\ell_t \geq \ell_\nu \geq \tilde{F}(x_\nu^{\mathrm{g}}) = \tilde{F}(x_{\min(t,\nu)}^{\mathrm{g}})$.

For $W = 1$, $D = 0$ and $\mathcal{C}_t$ is the single highest-scoring member of $\{x\} \cup \{\text{children of } x\}$ for the current state x, which is greedy's rule; with a common tie-breaking rule the two runs visit the same states, so $x_t^{\mathrm{g}} \in \mathcal{C}_t$ for all t and $\nu = \infty$. □

Remark 5 (The comparison cannot extend past ν)**.** *After displacement greedy draws from a state the beam no longer holds, so its children are never scored by the beam and elitism cannot retain them. Take $W = 2$, $D = 0$, $k = 2$ and cached scores. At $t = 1$ the seed (0.50) has children scoring 0.60 and 0.40; greedy moves to the 0.60 child and the beam keeps it and the seed. At $t = 2$ the seed draws children scoring 0.90 and 0.85, and the 0.60 state draws 0.61 and 0.30 (shared with greedy); greedy moves to 0.61, and the beam keeps $\{0.90, 0.85\}$, so $\nu = 2$. At $t = 3$ every child of the beam scores 0.10, while greedy's own draw from the 0.61 state scores 0.95. Then $\ell_3 = 0.90 < 0.95$, while (17) requires only $\ell_3 \geq 0.61$. Each step has positive probability under a proposer that conditions on the history, as Definition 13 allows.*

Remark 6 (Scored versus population comparison)**.** *The argument is carried out on the cached score $\tilde{F}(\cdot; \mathcal{D}_{\mathrm{valid}})$, which is the quantity both selectors act on, and it is exact there. The corresponding population statement $F^\star(\mathrm{leader}(\mathcal{C}_t)) \geq F^\star(x_{\min(t,\nu)}^{\mathrm{g}})$ does not follow pathwise, because a score ordering need not agree with the population ordering; it holds up to the uniform deviation of Theorem 7, i.e. with 2ε slack on the event of that theorem. We state the proposition on the scored objective for this reason.*

Remark 7 (Why Conjecture 1 resists this argument)**.** *The proof above compares the beam with greedy only up to the displacement step ν, and says nothing about how the beam's advantage grows with W. Showing that α_γ grows with W would require the W lineages to contribute independent progress, and the natural product-of-failure-probabilities argument does not apply: elitist selection couples the lineages, since which candidates survive depends on the scores of all of them jointly, and the reserved diversity slots induce further dependence through the shared quality gate $\tau_t = r\ell_t$. We therefore leave Conjecture 1 as a conjecture and rely on Proposition 5 wherever the comparison is needed.*

E.4 Elitist Monotonicity and Noisy No-Regression

This subsection characterizes what elitist selection buys over the children-only beam of ProTeGi. Throughout we use SPLICE's *cached-scoring convention* stated at the start of this section: each state x carries the regularized validation score $\tilde{F}(x; \mathcal{D}_{\mathrm{valid}})$ realized when x was first evaluated, not recomputed while it stays in the beam. This avoids the selection-on-noise bias that re-rolling would introduce, and its consequence for monotonicity is direct.

Lemma 1 is stated in the main text (p. 6).

We state Lemma 1 as a lemma rather than a theorem because it is a consequence of the definitions, not a claim about search dynamics. The proof is two lines: $\mathcal{C}_{t-1} \subseteq \mathcal{F}_t$ and scores are cached, so a maximum over a superset of a set whose values have not changed cannot fall. It therefore holds for *any* elitist selector over *any* proposal distribution, including an adversarial one. No property of prompt space is used and no experiment can falsify it.

It is nonetheless necessary: cached scoring is a design decision with a real cost, and this lemma justifies it. Later arguments use ℓ_t's monotonicity as a hypothesis. *The output rule* (Section 2) returns

$\mathrm{leader}(\mathcal{C}_T)$, and Lemma 1 is what makes this the best-scoring candidate of the run, so the returned prompt never scores below the seed; without monotonicity the final leader could fall below a candidate the run had already seen. *Adaptive annealing* (Proposition 1) sets the diversity gate at $\tau_t = r\,\ell_t$, so monotone ℓ_t is what makes τ_t non-decreasing and a candidate that falls below it *permanently* ineligible; if ℓ_t could fall the gate reopens and the argument collapses. And *the convergence rate* (Theorem 4) analyzes the leader score as a monotone accumulator of $\Delta_{\min}$ increments, which is what makes "improving steps required" a well-defined finite quantity. And *the comparison with greedy* (Proposition 5) uses it after the displacement step. One result we do *not* claim as a dependency: Theorem 6 evaluates its gate at a single step and never compares ℓ_t to ℓ_{t-1}, so it holds verbatim under re-evaluation.

The contrast with children-only beam search is still sharp: under $\mathcal{S}^{\mathrm{beam}}$ the previous leader is never retained, so a step in which all Wk children score below ℓ_{t-1} forces $\ell_t < \ell_{t-1}$. But Lemma 1 buys this on the *score estimate*, and the substantive monotonicity statement, the one that constrains behavior rather than fixing a definition, is the next result, which bounds regression of the beam's best *true* score under noise.

Theorem 2 is stated in the main text (p. 7).

Corollary 1 is stated in the main text (p. 7).

For $W = 4$, $k = 3$, $\delta = 0.05$ and independent batches ($c = 1$ at $\sigma^2 = 1/4$), Corollary 1 reads $n \geq \varepsilon^{-2}\ln(320) \approx 5.77\,\varepsilon^{-2}$: $n \geq 145$ at $\varepsilon = 0.2$, and $n \geq 5.8 \times 10^4$ at $\varepsilon = 0.01$. A shared batch doubles both. Read against our per-step training batches ($|B_t| \in \{10, 64, 216\}$), only BBH's reaches the $\varepsilon = 0.2$ requirement, and a batch of 64 buys $\varepsilon \approx 0.30$. So a per-step no-regression guarantee for the re-evaluation regime is coarse at every batch size we use, which is the quantitative reason SPLICE does not select on re-evaluated scores. We have not swept the batch size, so the corollary is a statement about the alternative design rather than one we test.

The two scoring conventions are a trade-off. It would be convenient to read Lemma 1 as the guarantee and Theorem 2 as a fallback for an ablation, but the two are not *guaranteed* versus *realistic* but two designs with opposite failure modes. *Cached scoring* gives deterministic non-regression on the score *estimate*, but the estimate can be stale: a candidate that drew a lucky batch keeps its inflated score for the rest of the run, so the leader may be a noise artifact. The guarantee is exact; the quantity guaranteed is imperfect. *Re-evaluation* tracks the true objective more closely, but re-rolls selection-on-noise at every step, so the leader can regress in truth, and Theorem 2 is what bounds that regression, with Corollary 1 naming the batch size that makes it small. Read together: *exact monotonicity on a possibly-stale estimate, or probabilistic monotonicity on the true score at a stated batch-size price.* Lemma 1 is the $n \to \infty$ corner of Theorem 2, where the bound vanishes and probabilistic non-regression becomes deterministic: one statement in two regimes rather than a result and its ablation.

The distinction matters when reading the experiments. SPLICE re-runs its beam on a fresh training batch every step to produce critiques, so a log of *those* scores can move down from one step to the next without contradicting Lemma 1: the lemma is about the cached selection score, not about fresh-batch re-evaluations. Score-progression traces should therefore be read as monotone in the cached score only, not as evidence of a monotone true-objective curve.

The rate counts progress in steps of a fixed size $\Delta_{\min} > 0$.

Definition 17 ($\Delta_{\min}$-improving proposal)**.** *A proposal $u \sim q_\phi(\cdot \mid x, H_t)$ is* $\Delta_{\min}$-improving from x *if* $\tilde{F}(\mathrm{Apply}(x, u)) \geq \tilde{F}(x) + \Delta_{\min}$.

Selection keeps any improvement, so smaller gains occur and are retained; they never lower the leader, and the definition only decides which proposals the rate counts. With a $0/1$ metric and $\lambda = 0$, every positive gain on a validation split of n examples is at least $1/n$, so $\Delta_{\min} = 1/n$ makes every improving proposal $\Delta_{\min}$-improving. Setting $\Delta_{\min}$ to SPLICE's `min_improvement` parameter (default 0.005) matches the threshold the algorithm uses for early stopping, but the analysis does not require that choice.

Before stating the rate we fix the quantity it is stated in, since the argument accumulates progress in the scored objective and the conclusion a reader wants is about the population one. Every ingredient

of the proof lives in the *scored* objective $\tilde{F}$ of Section 2: Definition 17 measures improvement in $\tilde{F}$, and Lemma 1 makes $\ell_t = \max_{\mathcal{C}_t} \tilde{F}$ non-decreasing. We therefore state the rate on $\tilde{F}$ and pay the transfer to $F^\star$ explicitly in Corollary 2, rather than assert a population conclusion the accumulation does not deliver. Write $\tilde{x}^\star \in \arg\max_{x \in \mathcal{X}} \tilde{F}(x)$.

Assumption 5 (Scored coverage)**.** *Fix a tolerance $\varepsilon > 0$, and let*

$$\tilde{p}^\star(\varepsilon) \;:=\; \min_{x\,:\,\tilde{F}(\tilde{x}^\star)-\tilde{F}(x)\,\geq\,\varepsilon} \;\Pr_{u\sim q_\phi(\cdot|x,H_t)} \big[u \text{ is } \Delta_{\min}\text{-improving from } x\big], \tag{19}$$

the minimum over states whose score is still ε below the best, uniformly over histories H_t, for the realized validation split. We assume $\tilde{p}^\star(\varepsilon) > 0$ at this one ε.

This is Assumption 1's minimum-competence condition read on the sublevel set of the objective the algorithm can see, with an improvement counted only when it reaches $\Delta_{\min}$. The two conditions range over different sets of states, since validation noise moves states into and out of the sublevel set, and neither implies the other. Theorem 3 needs the population form, because it reasons about the population suboptimality of parents; the rate below needs this one, because the leader is chosen by score.

Theorem 4 is stated in the main text (p. 7).

Corollary 2 (Population form, with the transfer priced)**.** *In the setting of Theorem 4, under Assumptions 2 and 3 and with a validation split of $n = |\mathcal{D}_{\text{valid}}|$ i.i.d. examples, let ε_n satisfy $2(|\mathcal{R}_T| + 1)\exp(-2n\varepsilon_n^2) \leq \delta/2$, with $\mathcal{R}_T$ the data-independent ball of Theorem 7. The quantities $\tilde{p}^\star$, K and p^{adv} depend on the realized validation split, and so does the horizon $T_{\text{req}} := (2K + 8\ln(2/\delta))/p^{\text{adv}}(\varepsilon)$. Then with probability at least $1-\delta$, either Assumption 5 fails at ε, or $T < T_{\text{req}}$, or*

$$F^\star(\text{beam leader}_T) \;\geq\; F^\star(x^\star) \;-\; \varepsilon \;-\; 2\varepsilon_n \;-\; \textstyle\sum_j \lambda_j R_j(x^\star). \tag{20}$$

Three of the four terms on the right of (20) are the price of reporting a population guarantee for a procedure that optimizes a regularized finite-sample score, and each is separately controllable: ε by the horizon, ε_n by the validation size, $\sum_j \lambda_j R_j(x^\star)$ by the regularization weights (for a single penalty, the term $\lambda\,\kappa(x^\star)$ that Theorem 5 isolates). The first two pull against each other: a longer horizon drives the ε requirement down but enlarges $\mathcal{R}_T$, and ε_n grows as $\sqrt{T\log(\Delta_{\mathcal{G}}+1)/n}$, so at a fixed validation size the population guarantee does not improve indefinitely with search. What no choice of T removes is the gap between $x^\star$ and $x^\star_{\text{all}}$, which is why (11) fixes the target at the best feasible prompt and Proposition 2 prices the rest.

One scope note. SPLICE early-stops after T_{pat} consecutive steps (five or ten in our runs) without an improvement exceeding `min_improvement`, so a run can terminate before the horizon Theorem 4 asks for. The bound therefore describes the unstopped process; early stopping trades the guarantee for budget exactly when p^{adv} is small enough to make the horizon impractical, and Lemma 1 is what makes that trade safe: the run exits with its monotone best-so-far rather than with a regression.

Two features of Theorem 4 matter. The exponent is k, *not* Wk: Theorem 3 guarantees a child that beats *its own parent*, whereas advancing the leader needs a child that beats *the leader*, and only the leader's own k draws are guaranteed to do that. The rate is therefore conservative, since a child of another lineage can also overtake the incumbent. And the dependence on δ is additive with explicit constants rather than an unspecified multiplicative $C\log(1/\delta)$, which is both sharper and checkable.

The bound has three empirical handles: the numerator scales with the initial gap, the denominator grows with k alone, and $\Delta_{\min}$ enters linearly. The k-dependence is the one the rate predicts. Because coverage enters through $\tilde{p}^\star(\varepsilon)$, the ε-dependence is explicit on both sides: tightening the target (smaller ε) enlarges the numerator's $(\tilde{h}_0 - \varepsilon)$ term and, because it enlarges the sublevel set over which $\tilde{p}^\star(\varepsilon)$ is a minimum, can only lower $p^{\text{adv}}(\varepsilon)$, so T grows on both counts. For small $\tilde{p}^\star$, $p^{\text{adv}} \approx k\,\tilde{p}^\star$, giving $T \propto 1/(k\,\tilde{p}^\star)$, linear in $1/\tilde{p}^\star$ with per-parent fan-out, not beam width, as the practitioner's compensating lever. When p^{adv} becomes very small the bound stays true and reports that the remaining gap is no longer reachable, and early stopping then ends the run with the leader it has.

Proofs.

Proof of Lemma 1. Let $x^{\mathrm{L}}_{t-1} \in \arg\max_{x \in \mathcal{C}_{t-1}} \tilde{F}(x; \mathcal{D}_{\mathrm{valid}})$, so $\tilde{F}(x^{\mathrm{L}}_{t-1}) = \ell_{t-1}$. Three facts combine.

(i) Retention in the frontier. By (6) the frontier is $\mathcal{F}_t = \mathcal{C}_{t-1} \cup \mathcal{N}_t \supseteq \mathcal{C}_{t-1}$, so $x^{\mathrm{L}}_{t-1} \in \mathcal{F}_t$. This is the step that fails for children-only beam search, where $\mathcal{F}_t = \mathcal{N}_t$.

(ii) Score stability. Under the cached-scoring convention, $\tilde{F}(x^{\mathrm{L}}_{t-1}; \mathcal{D}_{\mathrm{valid}})$ is the value realized when x^{L}_{t-1} was first evaluated and is not recomputed on retention. Hence its score in round t equals ℓ_{t-1} exactly.

(iii) Elitist selection. $\mathcal{S}^{\text{elit-div}}$ fills $\mathcal{E}_t$ with the top $W - D$ elements of $\mathcal{F}_t$ by score. Either $x^{\mathrm{L}}_{t-1} \in \mathcal{E}_t$, in which case $\ell_t \geq \tilde{F}(x^{\mathrm{L}}_{t-1}) = \ell_{t-1}$; or it is excluded, which by the definition of "top $W - D$ by score" requires at least $W - D$ elements of $\mathcal{F}_t$ with score at least $\tilde{F}(x^{\mathrm{L}}_{t-1})$, all of which are in $\mathcal{E}_t \subseteq \mathcal{C}_t$, so again $\ell_t \geq \ell_{t-1}$.

In both cases $\ell_t \geq \ell_{t-1}$, and t was arbitrary. □

Remark 8. *The argument uses only parent retention (i) and score stability (ii), and no property of q_ϕ or of the landscape, which is why Lemma 1 holds for any elitist selector over any proposal distribution, including an adversarial one, and why it is stated as a lemma in the body. It bounds the trajectory of the score* estimate*; the corresponding statement about the true objective is Theorem 2.*

Proof of Theorem 2. Here every member of $\mathcal{F}_t$ is re-evaluated at step t on n fresh examples and selection acts on $\hat{F}_n$, so the cached-scoring argument above is unavailable. Write $f^\star_t = \max_{x \in \mathcal{C}_t} F^\star(x)$ and let $\bar{x}_{t-1} \in \arg\max_{x \in \mathcal{C}_{t-1}} F^\star(x)$. Condition on everything up to the drawing of the step-t evaluation examples. The frontier $\mathcal{F}_t$ is then fixed (its parents were chosen at step $t-1$ and its children by the proposer), and $\bar{x}_{t-1} \in \mathcal{F}_t$ is fixed with it; only the step-t estimates remain random.

Suppose the regression event $\{f^\star_t < f^\star_{t-1} - \varepsilon\}$ occurs. Then $\bar{x}_{t-1}$ was not retained. The selector is elitist, so the top-scoring member x' of $\mathcal{F}_t$ is retained, and since $\bar{x}_{t-1} \in \mathcal{F}_t$ was not, $x' \neq \bar{x}_{t-1}$ and $\hat{F}_n(x') \geq \hat{F}_n(\bar{x}_{t-1})$; and every retained candidate has $F^\star(x') \leq f^\star_t < f^\star_{t-1} - \varepsilon$. Hence there is $x' \in \mathcal{F}_t \setminus \{\bar{x}_{t-1}\}$ with

$$F^\star(\bar{x}_{t-1}) - F^\star(x') \;>\; \varepsilon \qquad \text{and} \qquad \hat{F}_n(x') \;\geq\; \hat{F}_n(\bar{x}_{t-1}). \tag{21}$$

Writing $\eta(x) := \hat{F}_n(x) - F^\star(x)$, this gives $\eta(x') - \eta(\bar{x}_{t-1}) > \varepsilon$ for that x'.

Fix any $x' \in \mathcal{F}_t \setminus \{\bar{x}_{t-1}\}$ with population gap above ε. If the two candidates are scored on independent batches, each η is mean-zero sub-Gaussian with proxy σ^2/n (Lemma 2), their difference has proxy $2\sigma^2/n$ (Lemma 3), and $\Pr[\eta(x') - \eta(\bar{x}_{t-1}) > \varepsilon] \leq \exp(-n\varepsilon^2/4\sigma^2)$. If they share a batch, the difference is a mean of n i.i.d. per-example differences in $[-1, 1]$, and Hoeffding's inequality gives $\exp(-2n\varepsilon^2/2^2) = \exp(-n\varepsilon^2/2)$. Both are $\exp(-n\varepsilon^2/c)$ for the stated c.

Because $\mathcal{F}_t$ is fixed given the conditioning, a union bound over its at most $|\mathcal{F}_t| - 1$ members other than $\bar{x}_{t-1}$ is valid and gives Theorem 2. Taking the union over the retained beam $\mathcal{C}_t$ instead, as a smaller prefactor would suggest, would not be: $\mathcal{C}_t$ is chosen by the very estimates being bounded (Remark 12). Finally $|\mathcal{F}_t| \leq |\mathcal{C}_{t-1}| + k|\mathcal{C}_{t-1}| \leq W(1+k)$, and integrating over the conditioning preserves the bound. □

Proof of Corollary 1. Impose $W(1+k)\exp(-n\varepsilon^2/c) \leq \delta$, which implies the hypothesis of the corollary since $|\mathcal{F}_t| - 1 < W(1+k)$. Taking logarithms gives $n \geq (c/\varepsilon^2)\ln(W(1+k)/\delta)$, which is Corollary 1. With $W = 4$, $k = 3$, $\delta = 0.05$, $c = 1$: $n \geq \varepsilon^{-2} \ln 320$, i.e. 145 at $\varepsilon = 0.2$ and 5.77×10^4 at $\varepsilon = 0.01$. □

Proof of Theorem 4. Let $\tilde{h}_0 = \tilde{F}(\tilde{x}^\star) - \tilde{F}(x^{(0)})$ and $K := \lceil (\tilde{h}_0 - \varepsilon)/\Delta_{\min} \rceil$. Every quantity in this proof is the scored objective $\tilde{F}$; the transfer to $F^\star$ is Corollary 2.

Step 1: a bounded number of advances suffices. Call step t an *advance* if $\ell_t \geq \ell_{t-1} + \Delta_{\min}$. By Lemma 1 the leader score is non-decreasing, so it never gives back progress and the advances accumulate. Each advance raises the leader by at least $\Delta_{\min}$, so after K advances the leader has gained at least $K\Delta_{\min} \geq \tilde{h}_0 - \varepsilon$, leaving $\tilde{F}(\tilde{x}^\star) - \ell_t \leq \varepsilon$; that is, K advances suffice for ε-optimality in $\tilde{F}$. Note that K is a ratio of two quantities in the same units, which is what makes it the right count; dividing a population gap by a scored increment would not be.

Step 2: each step advances with probability at least p^{adv}. Let $t_\varepsilon := \min\{t : \tilde{F}(\tilde{x}^\star) - \ell_t \leq \varepsilon\}$ be the first ε-optimal step, and fix $t \leq t_\varepsilon$. Let x^{L}_{t-1} denote the current leader; since $t \leq t_\varepsilon$ it satisfies $\tilde{F}(\tilde{x}^\star) - \tilde{F}(x^{\mathrm{L}}_{t-1}) \geq \varepsilon$, so it lies in the sublevel set of (19). The leader contributes k of the expansion's draws. Each is $\Delta_{\min}$-improving from x^{L}_{t-1} (Definition 17) with probability at least $\tilde{p}^\star(\varepsilon)$ by (19), and by (6) they are i.i.d. given the history, so

$$\Pr\big[\text{some child of the leader beats it by at least } \Delta_{\min}\big] \;\geq\; 1 - \big(1 - \tilde{p}^\star(\varepsilon)\big)^k \;=:\; p^{\mathrm{adv}}(\varepsilon).$$

Any such child lies in $\mathcal{F}_t$ and scores at least $\ell_{t-1} + \Delta_{\min}$, so by the elitist argument of Lemma 1 the new leader satisfies $\ell_t \geq \ell_{t-1} + \Delta_{\min}$: the step is an advance. The bound holds conditionally on the entire past, since $\tilde{p}^\star(\varepsilon)$ is a uniform lower bound over states in the sublevel set and over histories.

Step 3: Chernoff, with the indicators patched after the target is met. Let I_t indicate that step t is an advance. Step 2 bounds its conditional success probability only for $t \leq t_\varepsilon$, so set $I'_t := I_t\,\mathbf{1}\{t \leq t_\varepsilon\} + \mathbf{1}\{t > t_\varepsilon\}$. The event $\{t \leq t_\varepsilon\}$ is fixed by the past at step t, since it says the target was not met by step $t-1$, so every I'_t has conditional success probability at least $p^{\mathrm{adv}}(\varepsilon)$ given the past, and Lemma 4 with $p = p^{\mathrm{adv}}(\varepsilon)$ applies to $N'_T := \sum_{t \leq T} I'_t$:

$$T \;\geq\; \frac{2K + 8\ln(1/\delta)}{p^{\mathrm{adv}}(\varepsilon)} \quad\Longrightarrow\quad \Pr[N'_T < K] \;\leq\; \delta.$$

On $\{t_\varepsilon > T\}$ every I'_t equals I_t, so N'_T is the number of advances, which is below K by Step 1. Hence $\Pr[t_\varepsilon > T] \leq \Pr[N'_T < K] \leq \delta$. On $\{t_\varepsilon \leq T\}$ the target is met at t_ε and kept to T by Lemma 1, since $\ell_T \geq \ell_{t_\varepsilon}$. So $\tilde{F}(\mathrm{leader}_T) \geq \tilde{F}(\tilde{x}^\star) - \varepsilon$ with probability at least $1 - \delta$, which is Theorem 4. □

Proof of Corollary 2. Write $\hat{x} := \mathrm{leader}_T$ and $n := |\mathcal{D}_{\mathrm{valid}}|$. Apply Theorem 7(i) to the set $\mathcal{U} := \mathcal{R}_T \cup \{x^\star\}$. It is fixed independently of $\mathcal{D}_{\mathrm{valid}}$: $\mathcal{R}_T$ is determined by the edit graph and the seed, and $x^\star$ by $F^\star$ and $\mathcal{X}$. In the setting of Theorem 4 scores are cached per state, so each member of $\mathcal{U}$ carries one score and the factor $T+1$ of Theorem 7(ii) does not arise. The set contains $\hat{x}$, since $\hat{x} \in \mathcal{H}_T \subseteq \mathcal{R}_T$ by Theorem 7(ii). Using the history itself as the union set would not be valid, for the reason Remark 12 gives. By hypothesis $2(|\mathcal{R}_T| + 1)\exp(-2n\varepsilon_n^2) \leq \delta/2$, so with probability at least $1 - \delta/2$,

$$|\hat{F}_n(x) - F^\star(x)| \;\leq\; \varepsilon_n \qquad \text{for every } x \in \mathcal{R}_T \cup \{x^\star\}. \tag{22}$$

Conditional on the validation split and the scores, Theorem 4 at confidence $\delta/2$, which is what the $8\ln(2/\delta)$ term provides, applies whenever Assumption 5 holds and $T \geq T_{\mathrm{req}}$. So the event that both hold and the conclusion of Theorem 4 fails has probability at most $\delta/2$, and with the failure of (22) the total is at most δ. The hypothesis and T_{req} depend on the split, which is why the corollary is stated as this joint bound rather than conditionally on the hypothesis. Outside the two failure events, either the hypothesis fails, or $T < T_{\mathrm{req}}$, or three inequalities compose. First, since every $R_j \geq 0$ and $\hat{x} \in \mathcal{R}_T$,

$$F^\star(\hat{x}) \;\geq\; \hat{F}_n(\hat{x}) - \varepsilon_n \;=\; \tilde{F}(\hat{x}) + \textstyle\sum_j \lambda_j R_j(\hat{x}) - \varepsilon_n \;\geq\; \tilde{F}(\hat{x}) - \varepsilon_n.$$

Second, Theorem 4 gives $\tilde{F}(\hat{x}) \geq \tilde{F}(\tilde{x}^\star) - \varepsilon$, and $\tilde{x}^\star$ maximizes $\tilde{F}$ over $\mathcal{X} \ni x^\star$, so $\tilde{F}(\hat{x}) \geq \tilde{F}(x^\star) - \varepsilon$. Third,

$$\tilde{F}(x^\star) \;=\; \hat{F}_n(x^\star) - \textstyle\sum_j \lambda_j R_j(x^\star) \;\geq\; F^\star(x^\star) - \varepsilon_n - \sum_j \lambda_j R_j(x^\star)$$

by (22). Chaining the three gives (20). □

Remark 9 (Which term is irreducible)**.** *Of the three penalties in* (20)*, ε shrinks with the horizon, ε_n with the validation size, and $\sum_j \lambda_j R_j(x^\star)$ with the regularization weights, so each can be*

driven down by a choice the practitioner controls. The quantity that no horizon reduces is the one (20) *does* not *contain: the gap between $x^\star$ and $x^\star_{\mathrm{all}}$. Stating the corollary against $x^\star$ rather than $x^\star_{\mathrm{all}}$ is therefore not a weakening for convenience: against $x^\star_{\mathrm{all}}$ the conclusion would be unreachable at every T whenever freezing costs more than ε, and* (19)*'s sublevel set would contain states with no improving neighbor, making the hypothesis false as well. Proposition 2 is where that gap is addressed, and it is addressed as a price rather than as a rate.*

Remark 10 (Why the rate carries k and not Wk)**.** *Step 2 uses only the k draws taken from the leader, because those are the ones whose improvement is guaranteed to move ℓ_t: a child improving on a non-leader parent need not exceed ℓ_{t-1} (Remark 2). The resulting rate $p^{\mathrm{adv}} = 1 - (1 - \tilde{p}^\star)^k$ is therefore conservative: children of other lineages can also exceed the incumbent leader, and nothing in the bound credits them. It should not be replaced by the reachability rate $p^{\mathrm{step}} = 1 - (1 - p^\star)^{|\mathcal{C}_{t-1}|k}$ of Theorem 3, which bounds a different event (some child beats its own parent) under a different coverage condition; using it would overstate what reachability supports, so we keep both quantities and label which is which, in the theory and in Section F.4.1.*

E.5 Diversity Preservation and the Quality Gate

Monotonicity describes the leader score but says nothing about the beam's *lineage structure*. Under pure elitism ($D = 0$) the leader climbs while all beam members may become descendants of one early winner (beam collapse).

Definition 18 (Beam collapse)**.** *Let* $\mathrm{anc}(x) := \mathrm{lineage}(x) \backslash \{\mathrm{id}(x)\}$ *denote the proper ancestors of x, and* $\mathrm{anc}^\circ(x) := \mathrm{anc}(x) \setminus \{\mathrm{id}(x^{(0)})\}$ *those other than the seed. A beam $\mathcal{C}$ with at least two members is* collapsed *if every pair of distinct members shares an ancestor other than the seed, that is, if no $x \neq x'$ in $\mathcal{C}$ have* $\mathrm{anc}^\circ(x) \cap \mathrm{anc}^\circ(x') = \emptyset$.

The seed has to be excluded, because every candidate after the first step descends from it. That is also the subtlety in Algorithm 3: its Tier 1 and Tier 2 tests check disjointness of anc, which *includes* the seed, so a candidate other than the seed can pass them only while the seed is the sole elite. The selector we ran therefore enforces diversity through its third tier (a distinct immediate parent) and through the backfill of Algorithm 2, and the guarantee below is stated for that mechanism. A selector that applied the same tests to anc° would enforce the stronger, lineage-level property; we state what it would guarantee but did not run it.

Theorem 6 (Diversity preservation)**.** *Let $D \geq 1$, $r \in [0, 1]$, and let $\mathcal{E}_t$ be the elite slots of Algorithm 2.*

(i) (As run.) *If $\mathcal{F}_t$ contains $c^\star$ with $\tilde{F}(c^\star) \geq r \cdot \ell_t$ whose parent is not the parent of any elite, then $\mathcal{C}_t$ contains a member scoring at least $r \cdot \ell_t$ whose parent is not the parent of any elite.*

(ii) (Lineage-level variant.) *If the Tier 1–2 tests of Algorithm 3 use* anc° *in place of* anc*, and $\mathcal{F}_t$ contains $c^\star$ with $\tilde{F}(c^\star) \geq r \cdot \ell_t$ and $\mathrm{anc}^\circ(c^\star) \cap \mathrm{anc}^\circ(e) = \emptyset$ for every elite e, then $\mathcal{C}_t$ is not collapsed whenever it has at least two members.*

The guarantee is *conditional*: the selector retains a distinct-parent (or, in the variant, lineage-disjoint) competitive candidate *when one exists*; generating one is the proposer's job. Part (i) is also weak by design. Two members with different parents may still share a recent ancestor, so it rules out the beam narrowing to one parent's children, not to one lineage. That is the property the D-sweep of Section 4 exercises, and its effect there is large; whether the lineage-level variant would add to it is untested.

Proposition 1 is stated in the main text (p. 6).

Proposition 1 is what an annealing reading of the gate rests on, and it is all that is proved: the gate rises with the leader, so a candidate that falls below it is never readmitted. The reading itself is that early on ℓ_t is low and many candidates qualify for the diversity slots, while late the gate admits only competitive ones, a shift from exploration to exploitation with no temperature parameter. That shift is

not a theorem, because eligibility compares a candidate with the leader, $\tilde{F}(c) \geq r\,\ell_t$, and whether fewer candidates qualify later depends on how new children score relative to a rising leader, which the selector does not control; children whose scores rise with the leader stay eligible. The exclusion is per candidate, not per lineage: a lineage whose current members fall below the gate can re-enter through a new, higher-scoring child. With $r = 0$ the gate is inactive and the proposition is empty; Section F.1 records which runs set $r > 0$. We do not report a lineage-count trajectory or an r-sweep, so the schedule is neither proved nor measured.

Proofs.

Proof of Theorem 6. Write $\mathcal{E}_t$ for the elites, $P = \{\mathrm{parent}(e) : e \in \mathcal{E}_t\}$ for their parents, and $\mathcal{Q}_t = \mathcal{F}_t \setminus \mathcal{E}_t$ for the pool PICKDIVERSE (Algorithm 3) scans in descending score order. Since $D \leq W - 1$, $\mathcal{E}_t \neq \emptyset$, and since the elites are the top of the frontier, $\ell_t = \max_{\mathcal{E}_t} \tilde{F}$, so the gate is $\tau_t = r\ell_t$. If Algorithm 2 takes its degenerate branch ($|\mathcal{F}_t| \leq W$ or $\mathcal{N}_t = \emptyset$) it returns the top $\min(W, |\mathcal{F}_t|)$ of the frontier; in the first case that is all of $\mathcal{F}_t$, which contains $c^\star$ and an elite and so the claim holds directly, and we assume the main branch below.

Part (i). $c^\star \notin \mathcal{E}_t$, since an elite's parent lies in P by definition. So $c^\star \in \mathcal{Q}_t$, and because the scan is in descending score order and $\tilde{F}(c^\star) \geq \tau_t$, no **break** occurs before $c^\star$ is reached. If Tier 1 or Tier 2 returns some c', then $\mathrm{anc}(c')$ is disjoint from every elite's anc; every non-seed candidate has the seed in anc, and at least one elite is non-seed whenever the beam has advanced, so c' is the seed, whose parent is undefined and therefore not in P. Otherwise Tier 3 reaches $c^\star$, which satisfies $\mathrm{parent}(c^\star) \notin P$, so it returns $c^\star$ or an earlier c'' with $\mathrm{parent}(c'') \notin P$. In every case the first diversity pick has a parent outside P and, since the scan stops at the gate, a score of at least $\tau_t = r\ell_t$.

Part (ii). If $c^\star \in \mathcal{E}_t$, the hypothesis applied to $e = c^\star$ gives $\mathrm{anc}^\circ(c^\star) = \emptyset$, so $c^\star$ has disjoint non-seed ancestry from every other member of $\mathcal{C}_t$, and the beam is not collapsed as soon as it has a second member, which the statement assumes. (The selector returns a single candidate only when every non-elite candidate falls below the gate and $W - D = 1$.) Otherwise $c^\star \in \mathcal{Q}_t$, it clears the gate, and it passes Tier 2 on anc°, so the scan returns $c^\star$ or an earlier pick passing Tier 1 or Tier 2, either of which has anc° disjoint from every elite's. That pick and any elite form the required pair, so $\mathcal{C}_t$ is not collapsed in the sense of Definition 18. □

Remark 11 (The guarantee is conditional, and on the proposer)**.** *Both hypotheses are about the* frontier*, not the selector: the theorem says that a competitive diverse candidate is retained* when one exists*. If every proposal descends from the current leader, no such $c^\star$ exists and the statement is vacuous; generating diverse candidates is the proposer's responsibility. Two further points about what ran. The backfill step of Algorithm 2 fills any slot* PICKDIVERSE *leaves empty with the best remaining candidate above the gate, so a D-slot arm is not "$W - D$ elites plus D diversity picks" but at most that many diversity picks with the rest chosen by score. And the theorem gives one diverse pair, not $D + 1$: a pairwise-diversity claim for $D \geq 2$ would need a hypothesis on the frontier at every slot, which we do not state.*

Proof of Proposition 1. By Lemma 1, $\ell_t \geq \ell_{t-1}$; since $r > 0$, multiplying through gives $\tau_t = r\ell_t \geq r\ell_{t-1} = \tau_{t-1}$, so the gate sequence is non-decreasing.

For the exclusion claim, suppose $\tilde{F}(x) < \tau_t$ for some candidate x and step t. With scores cached per state $\tilde{F}(x)$ is a fixed number, unchanged at every later step, while $\tau_s \geq \tau_t$ for all $s \geq t$ by the first part. Hence $\tilde{F}(x) < \tau_s$, and x fails the gate of Algorithm 3 at every $s \geq t$. The claim is about states only: new children of the same lineage are new states with new scores and may clear the gate. Without the per-state cache the claim holds per evaluation: a state re-proposed after leaving the beam is scored afresh and may clear the gate. Caching is essential: under re-evaluation $\tilde{F}(x)$ is re-drawn each step and a candidate below the gate can re-enter, which is the regime Theorem 2 governs. □

E.6 Generalization, Regularization, and the Price of Freezing

Three results close the analysis: a uniform-convergence bound relating empirical and population scores across the search (Theorem 7), the bias of length regularization (Theorem 5), and the approximation penalty of restricting search to mutable blocks (Proposition 2).

Theorem 7 (Uniform convergence over the search history)**.** *Let $\mathcal{H}_T := \bigcup_{t=0}^{T}(\mathcal{C}_t \cup \mathcal{N}_t)$ be the evaluated history, let $\mathcal{R}_T := \{x \in \mathcal{X} : d_{\mathcal{G}}(x^{(0)}, x) \le T\}$ be the radius-T ball of the edit graph around the seed, and let $\Delta_{\mathcal{G}}$ be the maximum degree of $\mathcal{G}$, so that $|\mathcal{R}_T| \le \min\{|\mathcal{X}|, (\Delta_{\mathcal{G}}+1)^T\}$. Under Assumption 3 and a validation split $\mathcal{D}_{\text{valid}}$ of n i.i.d. examples:*

(i) for any set $\mathcal{U} \subseteq \mathcal{X}$ fixed independently of $\mathcal{D}_{\text{valid}}$,

$$\Pr\left[\sup_{x\in\mathcal{U}} |\hat{F}_n(x) - F^\star(x)| > \varepsilon\right] \;\le\; 2\,|\mathcal{U}|\,\exp(-2n\varepsilon^2); \tag{23}$$

(ii) under Assumption 2, $\mathcal{H}_T \subseteq \mathcal{R}_T$ on every trajectory. Let $\mathcal{L}_T$ be the set of all evaluations the run makes up to step T, each a state x with the score $\hat{F}_n$ it received. Deduplication allows at most one evaluation of a state per step, so no state is evaluated more than $T+1$ times, and

$$\Pr\left[\sup_{(x,\hat{F}_n)\in\mathcal{L}_T} |\hat{F}_n - F^\star(x)| > \varepsilon\right] \;\le\; 2\,(T+1)\,|\mathcal{R}_T|\,\exp(-2n\varepsilon^2). \tag{24}$$

If every evaluation of a state returns the same score (a deterministic system agent, or scores cached per state), the factor $T+1$ can be dropped.

The two parts differ in exactly the way that matters here. The history $\mathcal{H}_T$ is *not* fixed independently of $\mathcal{D}_{\text{valid}}$: validation scores decide which parents survive, and the proposer conditions on those scores (Section 2), so which candidates exist at all depends on the selection split. A union bound over a set chosen with the data does not control the deviation on that set. If many candidates of equal population utility are scored and only the luckiest are expanded, the supremum over what was kept is the maximum of many deviations, not of a few. So part (i) applied to $\mathcal{H}_T$ as if it were fixed (which gives a half-width scaling as $\sqrt{\log|\mathcal{H}_T|}$, with $|\mathcal{H}_T| \le 1+TW(1+k)$) is not a guarantee for SPLICE. Part (ii) is, because $\mathcal{R}_T$ is a data-independent superset. The price is that it is loose: $\log|\mathcal{R}_T|$ grows linearly in T rather than logarithmically in the budget, and at the budgets of Section F.1 the guarantee exceeds 20 pp, wider than any margin this paper reports, so it is correct but not informative at our scale. Closing the gap between the two parts with an adaptive-data-analysis bound [7, 22, 67] is open.

Two points keep this from reaching further than it does. First, it is a *selection* guarantee, not a *reporting* one. Reported benchmark numbers are single evaluations on the untouched $\mathcal{D}_{\text{test}}$; $\hat{x}$ is a function of $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{valid}}$ only, so it is independent of the test sample and the reported number needs no union bound and no adaptivity correction. Adaptivity can make SPLICE select a worse prompt than it believes it has; it cannot make the reported test score of that prompt optimistic. Second, three properties of SPLICE plausibly narrow the adaptive channel, though none of them proves a bound. A state in the beam is never re-scored; a state that left the beam and is re-proposed is scored again, which with a stochastic system agent is a limited form of re-querying the validation split, and the factor $T+1$ in part (ii) prices it. Retention is a top-W cut, so validation enters selection through a ranking rather than through its full numerical detail. And the proposer never sees a validation *example*: it sees the scalar score $\tilde{F}(x)$ of its parent and critiques computed from $\mathcal{D}_{\text{train}}$ batches. Any residual selection bias then appears as a val–test gap on the returned prompt, which Section F.1 reports.

Theorem 5 is stated in the main text (p. 7).

The penalty for regularizing is at most λ times the cost of the unregularized optimum; the bound is tight at $\lambda = 0$ (no bias) and linear in λ. A λ-sweep reported naively would not be evidence for anything: the proof rearranges two $\arg\max$ definitions and uses only $\kappa \ge 0$, so the bound holds for any non-negative penalty on any search space and *no experiment can violate it*. It is also loose by construction, since the right-hand side is evaluated at the unregularized optimum. The growth the

regularizer corrects is an empirical property of the proposer's inductive bias in the additive-proposal regime rather than a theorem.

Proposition 2 is stated in the main text (p. 7).

The diminishing-returns hypothesis is not implied by γ-submodularity. The submodularity ratio bounds the immutable blocks' marginals *at* L, the part of $x^\star_{\text{all}}$'s commit set that leaves the frozen blocks at baseline, while Proposition 2 evaluates them *at the baseline* $x^{(0)}$, and Definition 16 says nothing about comparing the two (Remark 14). The hypothesis holds automatically at $\gamma = 1$, and in general it formalizes the reading below: contract blocks do not become *more* worth editing once the mutable blocks are optimized. Without it the bound still holds with the marginals taken at L, (26). The fixed ordering is needed because the ground sets of (13) are defined at $\pi^{(0)}$: an all-mutable optimum at another ordering has no commit set in E^{all}, and pricing it would require ordering moves in the ground set.

Under the diminishing-returns hypothesis, the price of freezing block i is at most its best single-block marginal gain *over the variants it would have admitted*, summed over frozen blocks and inflated by $1/\gamma$. When immutable blocks are safety constraints, format contracts, or tool schemas (whose marginal utility is *by design* small, because they constrain behavior rather than improve the metric), the right-hand side is small, and so is the price of selective mutability.

That reading is qualitative. Both factors on the right-hand side are outside what a deployment can measure: γ is the quantity we cannot estimate per benchmark (Section 5), and $\tilde{\mathcal{V}}_i$ is unobservable by construction, since it describes a search that the contract forbids running. We therefore do not report a realized-to-bound ratio for Proposition 2, and no experiment in this paper certifies it numerically. What Section F.2 does instead is measure a proxy for the left-hand side, the score gap between the returned prompts, comparing full SPLICE against an all-mutable variant, so it tests the conclusion rather than either hypothesis; it also reports the case the bound is least flattering in, where a frozen clause carries a large marginal gain by construction.

Scope. These results are stated at the granularity at which each component is defined. We do not attempt a single end-to-end sample-complexity bound composing every component: it would multiply the success probability of Theorem 3, the bias of Theorem 5, the terminal quality of the idealized greedy of Proposition 4, and the slack of Theorem 7 into constants no sharper than their product. The component-wise statements are more informative: each concerns one component, and an improvement to any one of them carries over to the composition.

Proofs.

Proof of Theorem 7. Part (i). Fix $x \in \mathcal{U}$. By Assumption 3 the per-example scores $r(\mathbf{x}, M(\mathbf{x}; \text{render}(x)), \mathbf{y})$ lie in $[0,1]$, and the validation split $\mathcal{D}_{\text{valid}}$ consists of n i.i.d. draws, so $\hat{F}_n(x)$ is an average of n independent $[0,1]$-valued variables with mean $F^\star(x)$ by (2). Lemma 2 gives

$$\Pr\big[|\hat{F}_n(x) - F^\star(x)| > \varepsilon\big] \;\le\; 2\exp(-2n\varepsilon^2).$$

This step uses that x is fixed before $\mathcal{D}_{\text{valid}}$ is drawn; it is the only place independence enters. Because $\mathcal{U}$ is fixed in the same sense, a union bound over its $|\mathcal{U}|$ members gives (23). The union bound needs no independence across members, so sharing one $\mathcal{D}_{\text{valid}}$ across all of them is harmless.

Part (ii). By Assumption 2, every child is $\text{Apply}(x, u)$ with u a primitive edit or the no-op, so it lies at edit-graph distance at most one from its parent. Parents at step t belong to $\mathcal{C}_{t-1} \subseteq \mathcal{H}_{t-1}$, and $\mathcal{C}_0 = \{x^{(0)}\}$, so by induction every member of $\mathcal{H}_T$ is within distance T of $x^{(0)}$: $\mathcal{H}_T \subseteq \mathcal{R}_T$ on every trajectory, whatever the data. The ball $\mathcal{R}_T$ is determined by $\mathcal{G}$ and $x^{(0)}$ alone, so it is fixed independently of $\mathcal{D}_{\text{valid}}$.

A state may be evaluated more than once: deduplication compares a proposal only with the current beam and the step's other proposals, so a state that left the beam can be re-proposed and re-scored. Deduplication removes duplicates within a step, so a state is evaluated at most once per step and at most $T+1$ times in all. Each evaluation draws fresh model randomness on the same $\mathcal{D}_{\text{valid}}$. Realize that randomness in advance: for every $x \in \mathcal{R}_T$ and $j \le T+1$, draw independently the model outputs the j-th evaluation of x will use, and let $\hat{F}_n^{(j)}(x)$ be the resulting score. This table has the same law

as the scores the run produces, and it does not depend on the run's choices, so each $\hat{F}_n^{(j)}(x)$ is an average of n independent $[0,1]$-valued scores with mean $F^\star(x)$, and Lemma 2 applies to it as in part (i). Every evaluation in $\mathcal{L}_T$ is one entry of the table, so

$$\Pr\Big[\sup_{(x,\hat{F}_n)\in\mathcal{L}_T} |\hat{F}_n - F^\star(x)| > \varepsilon\Big] \;\le\; \Pr\Big[\max_{x\in\mathcal{R}_T,\, j\le T+1} |\hat{F}_n^{(j)}(x) - F^\star(x)| > \varepsilon\Big]$$
$$\le\; 2(T+1)|\mathcal{R}_T|\exp(-2n\varepsilon^2),$$

which is (24). If every evaluation of a state returns the same score (as when scores are cached per state), the table has one distinct entry per state and the union runs over $\mathcal{R}_T$ alone. For the cardinality, every state has at most $\Delta_\mathcal{G}$ neighbors, so the ball holds at most $\sum_{j=0}^{T}\Delta_\mathcal{G}^j \le (\Delta_\mathcal{G}+1)^T$ states, and trivially at most $|\mathcal{X}|$. □

Remark 12 (Why the history itself cannot be the union set)**.** *It is tempting to bound the deviation on $\mathcal{H}_T$ by $2|\mathcal{H}_T|\exp(-2n\varepsilon^2)$, since $|\mathcal{H}_T| \le 1 + TW(1+k)$ deterministically. That does not follow. A union bound controls the probability that* some member of a fixed set *deviates; it says nothing about a set whose membership is chosen after looking at the deviations. A two-line instance shows the gap. Let J candidates share the same population utility, score all of them on one $\mathcal{D}_{\text{valid}}$, and keep the $N \ll J$ with the highest scores. The kept set has cardinality N, but the largest deviation in it is the largest of all J, so its tail is governed by J, not N.* SPLICE *is a milder version of the same mechanism: scores decide which parents are expanded, and the proposer conditions on the parent's score, so the history is partly a function of validation noise.*
This is why Theorem 7 states the history bound through the data-independent ball $\mathcal{R}_T$ rather than through $\mathcal{H}_T$. The cardinality-$|\mathcal{H}_T|$ figure remains useful as a calibration (it is exactly what a procedure that fixed its candidates in advance would get, so no worst-case adaptive accounting can do better), but it is not itself a guarantee for SPLICE*. On SST-5 the two are 10.3 pp and above 24 pp respectively (Section F.1); the difference is the price of adaptivity under the crudest valid accounting.*

Proof of Theorem 5. Let $x^\star \in \arg\max_\mathcal{X} F^\star$ and $x_\lambda^\star \in \arg\max_\mathcal{X} F_\lambda^\star$ where $F_\lambda^\star = F^\star - \lambda\kappa$ with $\lambda \ge 0$ and $\kappa \ge 0$. By optimality of $x_\lambda^\star$ for $F_\lambda^\star$ and feasibility of $x^\star$,

$$F^\star(x_\lambda^\star) - \lambda\kappa(x_\lambda^\star) \;\ge\; F^\star(x^\star) - \lambda\kappa(x^\star).$$

Rearranging,

$$F^\star(x^\star) - F^\star(x_\lambda^\star) \;\le\; \lambda\big(\kappa(x^\star) - \kappa(x_\lambda^\star)\big) \;\le\; \lambda\,\kappa(x^\star),$$

the last inequality because $\kappa(x_\lambda^\star) \ge 0$. This is Theorem 5. □

Remark 13 (The bound is unfalsifiable)**.** *The proof uses only the two* $\arg\max$ *definitions and $\kappa \ge 0$; it appeals to no property of $\mathcal{X}$, of q_ϕ, or of the task. The inequality therefore holds for any non-negative penalty on any search space, and no experiment can violate it.*

Proof of Proposition 2. Throughout, the ground set is E^{all} of (13), not E: the statement compares an optimum that may edit frozen blocks against one that may not, so the frozen coordinates must be present in the ground set for either the hypothesis or the conclusion to have content. On E the set S_{imm} constructed below is empty, every marginal in (26) vanishes, and the argument degenerates to $0 \le 0$.

Let $S^\star \subseteq E^{\text{all}}$ be the set of block-variant commits realizing $x_{\text{all}}^\star$, which exists because $\Pi = \{\pi^{(0)}\}$, let $S_{\text{imm}} := \{(i,v) \in S^\star : i \in I_{\text{imm}}\}$ be its entries on immutable blocks, and put $L := S^\star \setminus S_{\text{imm}} \subseteq E$. Then x_L leaves every immutable block at its baseline variant, so $x_L \in \mathcal{X}$ is feasible for the restricted search and therefore

$$g(L) \;\le\; F^\star(x_{\text{mut}}^\star) - F^\star(x^{(0)}). \tag{25}$$

Apply Definition 16 on $E' = E^{\text{all}}$ with this L and $S = S_{\text{imm}}$, which are disjoint by construction:

$$\gamma\big[g(L \cup S_{\text{imm}}) - g(L)\big] \;\le\; \sum_{e\in S_{\text{imm}}} \big[g(L\cup\{e\}) - g(L)\big].$$

If $g(L \cup S_{\mathrm{imm}}) = g(L)$ the left-hand side of (26) is at most zero by (25) and the claim is immediate; otherwise the pair is admissible in the minimum defining $\gamma(g; E^{\mathrm{all}})$. Since $L \cup S_{\mathrm{imm}} = S^\star$ and $g(S^\star) = F^\star(x^\star_{\mathrm{all}}) - F^\star(x^{(0)})$, dividing by $\gamma > 0$ and combining with (25) gives

$$F^\star(x^\star_{\mathrm{all}}) - F^\star(x^\star_{\mathrm{mut}}) \;\le\; g(S^\star) - g(L) \;\le\; \frac{1}{\gamma} \sum_{e \in S_{\mathrm{imm}}} \big[g(L \cup \{e\}) - g(L)\big], \tag{26}$$

which is the price of freezing expressed through the marginal gains of the immutable blocks *evaluated at* L, the mutable part of $S^\star$. This step uses only Assumption 4. Under the diminishing-returns hypothesis of Proposition 2, Corollary 3 converts the right-hand side to the baseline form Proposition 2, which completes the proof. □

Corollary 3 (Baseline form of the price of freezing)**.** *If in addition g has diminishing returns on immutable coordinates, i.e. $g(L \cup \{e\}) - g(L) \le g(\{e\}) - g(\emptyset)$ for every $e \in S_{\mathrm{imm}}$ and the above L (implied by the hypothesis of Proposition 2, since $L \subseteq E$ and $S_{\mathrm{imm}} \subseteq E^{\mathrm{all}} \setminus E$), then*

$$F^\star(x^\star_{\mathrm{all}}) - F^\star(x^\star_{\mathrm{mut}}) \;\le\; \frac{1}{\gamma} \sum_{i \in I_{\mathrm{imm}}} \max_{v \in \tilde{\mathcal{V}}_i} \big[F^\star(x^{(0)}[i{:=}v]) - F^\star(x^{(0)})\big], \tag{27}$$

which is Proposition 2.

Proof. Substituting the diminishing-returns inequality into (26) replaces each marginal at L by a marginal at $\emptyset$; then $g(\{(i,v)\}) - g(\emptyset) = F^\star(x^{(0)}[i{:=}v]) - F^\star(x^{(0)})$ by (14), and bounding each term by the best counterfactual variant of its block (the maximum over $\tilde{\mathcal{V}}_i$, which is the set the entries of S_{imm} are drawn from) and each block of S_{imm} by all of I_{imm} (using monotonicity, so the omitted terms are non-negative) gives (27). □

Remark 14 (What the extra condition costs, and why we state it)**.** *Equation Proposition 2, which Section E reads as a bound by each frozen block's best single-block gain* at the baseline*, is* (27)*, and it needs the diminishing-returns step of Corollary 3. That step does* not *follow from γ-submodularity alone: Definition 16 lower-bounds a sum of marginals at L by a joint gain, and says nothing about comparing a marginal at L to a marginal at $\emptyset$. It holds automatically at $\gamma = 1$ (true submodularity), and more generally it is the natural formalization of the claim the interpretation relies on, namely that contract blocks do not become* more *valuable to edit once the mutable blocks have been optimized. It is kept separate from Assumption 4 because it is a second structural assumption and the γ in* (27) *is already the quantity we cannot estimate per benchmark (Section 5). The unconditional statement is* (26)*, and the empirical check of Section F.2 measures the left-hand side directly, so it tests the conclusion rather than the hypothesis.*

Remark 15 (The right-hand side is not measurable, and why we keep it anyway)**.** *Both statements quantify over the counterfactual variant sets $\tilde{\mathcal{V}}_i$ of* (13)*, and these are unobservable in any setting where the proposition is worth invoking: recovering $\tilde{\mathcal{V}}_i$ means running the proposer against block i with the contract released, which is exactly what a contract forbids. Together with γ, this makes the right-hand side of Proposition 2 an object no experiment in this paper estimates.*

The counterfactual set is nonetheless the only choice that leaves the proposition with content. Taking the maximum over $\mathcal{V}_i$ would make the right-hand side computable, but trivially: Definition 3 gives $\mathcal{V}_i = \{v_i^{(0)}\}$ for every $i \in I_{\mathrm{imm}}$, so every term is $F^\star(x^{(0)}) - F^\star(x^{(0)}) = 0$ and the bound asserts that freezing is free, which Section F.2 refutes directly. What Proposition 2 supplies is therefore a shape *rather than a budget: the price of freezing is controlled by the marginal value of the frozen blocks. That is the claim Section E.6 makes, and Section F.2 probes it by constructing a frozen block whose marginal value is large by design.*

F Experimental Details

This section extends Section 4: protocols and splits, the full contract study, the ablations summarized there, capacity and cost, the judge-alignment application, and per-benchmark diagnostics.

F.1 Protocol and Reporting

THREE MODEL ROLES. Every run has three model roles, and keeping them apart is what the opening of this section promised. The *target* model is the one whose prompt is being optimized and on which the returned prompt is deployed; it is fixed by whichever published protocol we are reproducing. The *proposal agent* samples the k candidate edits per parent, and the *critique agent* produces the dual section-local gradients of (7). Proposal and critique are ordinarily distinct models: the same model both generating and consuming textual gradients tends toward agreement with its own reasoning rather than a corrective signal. The exception is AIME-2025, where both roles run on the same model and are separated only by temperature (0.9 for proposal, 0.0 for critique); we note it because it is the one setting where the echo-chamber argument does not apply. In some folds of the GPQA sweep and the judge task, the target is the same model as the proposer or critic.

THE STATE, CONCRETELY. Definition 5 is instantiated by a template of named sections. On SST-5 and AIME the prompt is $m = 5$ blocks (role, objective, approach, examples, and output format) rendered to a single XML stream, with the output-format block tagged `immutable` and the remaining four mutable. So $|I_{\mathrm{mut}}| = 4$, the frozen coordinate is the output contract, and $\mathcal{V}_i$ for that block is a singleton by construction rather than by instruction. The contract experiment of Section F.2 extends this template with a seven-clause contract block and varies exactly which blocks carry the tag.

SEARCH CONFIGURATION. Search parameters differ by benchmark, so there is no single default: beam width is 5 on SST-5 and AIME (4 in the seeded SST-5 sweeps) but 4 on BBH, GPQA-Diamond and HelpSteer2; the fan-out k ranges over 3–5; and the quality-gate ratio r of Proposition 1 is 0.7 on SST-5 and AIME and 0.9 on HelpSteer2, while the BBH and GPQA-Diamond configurations do not set it, so the implementation's default $r = 0$ applies and the gate is inactive there. Every benchmark number uses $\lambda = 0$. Every benchmark follows its source protocol exactly (target model, decoding settings, splits and metric). Unless an ablation varies them, proposals and critiques come from fixed proposal and critique models (proposal temperature 0.9, 1.0 on BBH; critique temperature 0.0), with AIME-2025 the exception noted above. The improvement threshold is $\Delta_{\min} = 0.003$ on SST-5 and AIME and 0.005 on BBH and GPQA. Momentum is enabled at window $w = 5$ throughout, the setting Section F.3.2 examines directly.

THREE SPLITS, THREE ROLES. Every run uses three disjoint i.i.d. splits, and both the guarantees and the reported numbers depend on which split does what. $\mathcal{D}_{\mathrm{train}}$ supplies a fresh batch B_t each step on which the current beam is re-run; those scores feed the critique and are never used for selection, retention, gating, stopping or output. $\mathcal{D}_{\mathrm{valid}}$ is fixed and used to score and select candidates, under the cached-scoring convention of Section 2: each candidate is scored once when it is created and carries that score while it stays in the beam, rather than being re-rolled. (A candidate that leaves the beam and is later re-proposed verbatim is scored again, since deduplication checks only the current beam.) This is what makes Lemma 1 operative rather than merely formal. $\mathcal{D}_{\mathrm{test}}$ is disjoint and never queried during search. The returned prompt is $\mathrm{leader}(\mathcal{C}_T)$, which under elitist selection is also the best-scoring candidate the run evaluated, so its validation score is optimistic and we report test accuracy separately.

REPORTING CONVENTION. We label the two accuracies explicitly, because the distinction carries the statistical argument below. *Val* is accuracy on $\mathcal{D}_{\mathrm{valid}}$: what the optimizer sees, and the quantity every selection decision is made against. *Test* is accuracy on $\mathcal{D}_{\mathrm{test}}$, a single evaluation of the single returned prompt. Unless a number is marked *Val*, it is test accuracy.

ADAPTIVE REUSE OF THE VALIDATION SPLIT. $\mathcal{D}_{\mathrm{valid}}$ is queried repeatedly across a run to rank candidates, so a single-candidate concentration bound does not cover *selection*, and we separate the two claims it might be asked to support. For *reporting*, the returned prompt $\hat{x}$ is a function of $\mathcal{D}_{\mathrm{train}}$, $\mathcal{D}_{\mathrm{valid}}$ and the optimizer's own randomness, so $\hat{x}$ is independent of the test sample and the reported number needs no union bound and no adaptivity correction. For *selection*, the correct statement is the uniform bound of Theorem 7, and it is worth being exact about which part of it applies. Bounding by the size of the evaluated history, $|\mathcal{H}_T| \le 1 + TW(1 + k)$, would give a 95% uniform half-width of

10.3 pp on SST-5 (n=500, W=5, k=3, T=50, so $|\mathcal{H}_T| \leq 1001$) against 6.1 pp for a single candidate. That figure is not a guarantee for SPLICE, because the history is chosen using the same validation scores it would be bounding (Remark 12); it is what a procedure that fixed its candidates in advance would get. The valid statement unions over the radius-T ball of the edit graph, which contains every history and does not depend on the data. Its half-width is $\sqrt{(\ln(2(T+1)/\delta) + T\ln(\Delta_{\mathcal{G}}+1))/2n}$, which at T=50 exceeds 24 pp for every maximum degree $\Delta_{\mathcal{G}} \geq 2$. So we have a correct selection guarantee that is uninformative at our budget, and an informative figure that is not a guarantee. We report both rather than let the second pass for the first.

That leaves the empirical check carrying the weight. Validation scores decide which parents survive, and the proposer conditions on its parent's score, so the history depends on validation noise. Three properties of SPLICE plausibly narrow that channel without bounding it: each candidate is scored once while it stays in the beam, retention is a top-W cut rather than a numerical reweighting, and the proposer sees scalar scores and $\mathcal{D}_{\text{train}}$ critiques but never a $\mathcal{D}_{\text{valid}}$ example. What those properties cannot do is stop the search from fitting validation noise, and on the one configuration where we can check, it does. The seeded SST-5 runs (the momentum and contract experiments) are the only seed-paired runs for which we report both a validation and a test score for the returned prompt: in the momentum control arm (Table 14) the leader scores 59.9 on validation and 56.3 on test, a 3.6 pp optimism, against a seed prompt that scores 54.0 and 56.2. So the search gained 5.9 pp on the split it selected on and 0.1 pp on the split it did not. That is selection-on-noise during search, and we report it as such. It does not bias the reported number, which is measured on $\mathcal{D}_{\text{test}}$ and is independent of how $\hat{x}$ was chosen; it does mean a validation curve in this paper should not be read as an estimate of held-out gain. The GPQA sweep has no test split, and its train–validation gaps (Figure 17 in Section F) are not evidence either way about validation optimism.

REPRODUCIBILITY. Prompt optimization is unusually sensitive to details that are easy to leave unstated (the seed prompt, decoding settings, parsing rules, model versions, splits, and candidate budget), so we state what the seed controls and what it does not.

The random seed governs batch draws and data ordering. It does *not* control sampling inside the language models: the proposal agent runs at non-zero temperature behind a black-box endpoint, and its draws are not reproducible from any seed we hold. Every spread we report across seeds is therefore *total* run-to-run variance (seed-controlled data sampling plus irreducible model sampling noise) rather than the seed-attributable component alone, which makes it a conservative estimate of stability: it can only overstate the variability a fixed-data rerun would show.

Table 8 reports the full SST-5 results alongside the published baselines from the two protocols. Under the [89] protocol (target: `gemma-3-27b-it`), SPLICE reaches 59.2% accuracy, +2.9 pp over the strongest published method (CAPO at 56.3). Under the [94] protocol (target: `deepseek-v3.2`), SPLICE reaches 57.4%, +0.8 pp over the strongest prior method (EvoPrompt at 56.6). The gap is larger on `gemma-3-27b`, whose protocol baseline is 44.6% against 56.0% for `deepseek-v3.2`, and the published methods cluster in 53.3–56.6% on both (GEPA's 42.0 aside). A 2.9 pp gap over CAPO corresponds to an error-rate reduction from 43.7% to 40.8%, a 6.6% relative reduction; the same gap on `deepseek-v3.2`'s saturated band would require a method that breaks out of the cluster entirely, which no published method in this comparison does.

On the [37] protocol (GPT-3.5-Turbo), SPLICE reaches 91.0% on ARC-Challenge and 83.09% on GSM8K (Table 2), clearing the strongest UniPrompt configuration by +0.5 pp and +0.7 pp respectively. The lead over the TextGrad / ProTeGi family is wider: +12.2 pp on ARC-Challenge over ProTeGi and +1.5 pp on GSM8K. Under the [1] protocol (target GPT-4.1 Mini; Table 3), SPLICE reaches 64.4% on AIME-2025 and 96.24% on PUPA (+5.1 pp over GEPA on AIME, within 0.22 pp of GEPA+Merge on PUPA). Under the [65] protocol (Claude-4.5-Haiku; Table 4), SPLICE reaches 92.6% on BBH (+0.7 pp over GEPA) and ties GEPA at 68.4% on GPQA-Diamond, +3.1 pp over the protocol's baseline prompt and +4.1 pp over ContraPrompt.

Under the accounting of Section 4, a margin of at most 0.5 pp either way is a match: ARC-Challenge (+0.5), GPQA-Diamond (0.0) and PUPA (−0.22). The GSM8K and BBH wins (+0.69, +0.70 pp) clear that threshold only narrowly and sit inside the range that differences in tuning, API version or prompt formatting could account for, since none of the five reproduced protocols reports seed variance for the baselines we compare against. The claims we rest on are the margins with headroom behind them (AIME-2025 and SST-5 on `gemma-3-27b`) and the absence of any loss beyond the threshold on any of the five targets.

Table 8: Benchmark results on SST-5 across two evaluation protocols. Gemma-3-27B numbers follow [89]; DeepSeek-V3.2 numbers follow [94]. "—" indicates the method was not run under the corresponding protocol. SPLICE is the test mean over three runs; baselines are as published. Bold marks the best score per column.

Method	Gemma-3-27B	DeepSeek-V3.2
Baseline	44.6	56.0
AutoDiff	55.7	—
OPRO	56.0	—
EvoPrompt	53.3	56.6
GEPA	42.0	—
CAPO	56.3	—
ProTeGi	—	56.4
BPO	—	56.4
SPLICE	**59.2**	**57.4**

F.2 Does the Constrained Formulation Buy Anything?

Frozen blocks are preserved by construction: the search space contains no variant of an immutable block, so reporting a 0% modification rate against a baseline's nonzero rate restates Definition 5 rather than measuring anything. The question that is not definitional is the one a practitioner actually faces: *when an optimizer is able to rewrite a contract, does it, and what does that cost?* Answering it needs a setting where the contract is worth breaking. We construct one.

THE CONTRACT. Table 9 gives the seven-clause contract, which instantiates all four classes of production artifact that motivate Section 2: a `record_sentiment` tool schema, safety text, a strict output envelope, and a business rule. The task is SST-5 and the target is `gemma-3-27b-it`. We take the deliberate trade of a synthetic-but-exact contract over a real deployed one: a production prompt would be unreproducible and would require an LLM judge to rule on whether a given rewrite counts as a violation, whereas here the frozen text is fixed and public, so detection is a regular expression and a parser and every rate we report is exact and recomputable.

The clause that carries the experiment is a compliance-routing rule of the kind found in deployed agents:

> POLICY-1. *If the snippet mentions "film" or "movie", do not classify it. Emit* `{"label": "ESCALATE"}`*; these require human review.*

The optimizer's metric is classification accuracy, so an escalation is not a sentiment label, scores zero, and *obeying* POLICY-1 forfeits score. We call this metric the *proxy*. The mis-specification is deliberate and, we would argue, typical: hand-written scoring functions routinely omit credit for a correct refusal. Table 10 gives the per-example credit under the proxy and under the contract-aware metric that scores the deployed objective, together with the ceilings each implies. With a 26% trigger rate the two are near-opposites on the examples POLICY-1 governs: the proxy caps a fully compliant prompt at 74% and the only route to 100% is total non-compliance. That ≈ 26 pp gap is the score on offer for breaking the contract, and it is the setup rather than an artifact of it.

WHY THIS CLAUSE AND NOT AN EASIER ONE. A contract that is costless to obey cannot demonstrate the value of protecting it: if compliance never reduces the metric, no optimizer has an incentive to break it and every method looks equally safe. We verified this concretely: several natural safety clauses in the same prompt (forbidding chain-of-thought, requiring the full five-point label scale) *improved* accuracy when obeyed, and no arm disturbed them. POLICY-1 satisfies three requirements simultaneously. It *can be obeyed*: the trigger is lexical, so compliance is a decision rather than a capability limit, and the unoptimized seed prompt (A0) complies on 96.5% of triggered test examples. (A semantically gated variant, "abstain when uncertain", was ignored outright and produced no measurable behavior.) It *is costly*: the rule fires on 26% of examples. And it *fails silently*: a missed escalation is well-formed, plausible output that raises no parser or schema error, so the optimizer receives no corrective signal. This is the class of failure that format validation cannot detect and that selective mutability is designed to guard against.

Contract class	Instantiation	Costs reward?
Tool schema	`record_sentiment`: enum-constrained label, `additionalProperties: false`	✗
Safety text	no input echo; no chain-of-thought; prompt-injection resistance	✗
Output contract	lone JSON envelope, no surrounding prose	✗
Business rule	POLICY-1 compliance routing	✓

Table 9: **The seven-clause contract, by artifact class.** The prompt instantiates all four classes of production artifact that motivate the constrained formulation of Section 2, rather than being a benchmark prompt with a safety sentence appended. The trade is deliberate: a prompt from a live deployment would be unreproducible and would require an LLM judge to rule on whether a given rewrite constitutes a violation, whereas here the frozen text is fixed and public, so detection is a regular expression and a parser and every rate in Table 11 is exact and recomputable. Only the business rule is reward-conflicted, and that asymmetry is the experiment: the other three classes were left untouched by *every* arm in *every* candidate. Optimizers do not vandalize indiscriminately; they trade away precisely the contracts that cost reward, and a practitioner cannot know in advance which those are.

Example	Output	Proxy	Contract
$i \in \mathcal{T}$	`ESCALATE`	0	**1**
$i \in \mathcal{T}$	label, correct	**1**	0
$i \in \mathcal{T}$	label, wrong	0	0
$i \in U$	label, correct	1	1
$i \in U$	label, wrong	0	0
$i \in U$	`ESCALATE`	0	0

A perfect classifier that…	Proxy (val)	Contract (test)
ignores POLICY-1	**100.0**	73.7
fully obeys POLICY-1	74.0	**100.0**

Table 10: **Left: per-example credit under the two scoring rules.** $\mathcal{T}$ is the set of POLICY-1-triggered examples (26.0% of $\mathcal{D}_{\text{valid}}$, 26.3% of $\mathcal{D}_{\text{test}}$) and U the remainder; each metric is the mean credit over all n examples. Two rows carry the design. Row 1 is the mis-specification: a correct refusal earns 0 under the proxy and 1 under the contract-aware metric, so the proxy silently punishes the behavior the contract mandates. Row 6 is the anti-gaming check: a spurious refusal earns 0 under both, so blanket escalation scores only $|\mathcal{T}|/n \approx 26\%$ rather than 100% and the contract-aware metric cannot be gamed by indiscriminate abstention. **Right: the resulting ceilings.** With a 26% trigger rate the two objectives are near-opposites on the examples POLICY-1 governs, so each metric's ceiling sits close to the other's floor. The proxy caps a fully compliant prompt at 74% and the only route to 100% is total non-compliance: that ≈26 pp gap is the reward on offer for breaking the contract, and it is the setup rather than an artifact of it.

ARMS AND PROTOCOL. Four arms share the contract text, hyperparameters, splits and objective, and differ *only* in whether the contract is mutable: **A1** SPLICE (immutable), **A2** all-mutable (identical blocks, editable), **A3** monolithic (the entire prompt as one editable block, which is the representation every prior method searches over), and **A4** soft-instructed (editable, plus an explicit "do not edit" directive). A3 and A4 both *request* preservation; only A1 enforces it, and that contrast is what separates a structural guarantee from an instruction. Splits follow Section F.1: fresh training batches produce critiques only, a fixed 300-example $\mathcal{D}_{\text{valid}}$ selects candidates under the proxy, and the 2,210-example $\mathcal{D}_{\text{test}}$ is queried once on the returned prompt under the contract-aware metric. The optimizer therefore cannot optimize the contract-aware objective (it never observes it), and the search proceeds exactly as a practitioner's would under a misspecified metric. Gold sentiment labels are unmodified; only the scoring rule differs, on the 26.3% of test examples POLICY-1 governs. Three paired seeds.

RESULTS. Table 11 reports the fate of POLICY-1 over every candidate the search produced; Table 12 reports accuracy and held-out compliance, and Figure 2 plots them. Modification rates are 0% for A1 by construction, 72.8% for A4, 88.2% for A3 and 98.2% for A2. The ordering *inverts* between the two metrics: every unconstrained arm beats SPLICE by roughly 10 pp on the proxy and loses by roughly 17 pp on the objective a practitioner deploys against, and escalation compliance on held-out test is 66.8% for A1 against 0–1.4% for the editable arms. The optimizer did exactly what it was asked, and the result is worse to deploy.

Four readings carry the experiment. *First, the arms are equally good classifiers.* Untriggered label accuracy backs out to ≈55.5% (A1) against ≈56.2% (A2), so the entire 17.1 pp gap is escalation

Arm	Cands.	Verbatim	Reworded	Deleted	Mod. rate
A0 seed (no optimization)	3	3	0	0	0.0
A1 SPLICE (contract frozen)	456	**456**	0	0	**0.0**
A2 all-mutable	336	6	107	**223**	98.2
A3 monolithic	372	44	**316**	12	88.2
A4 soft-instructed	408	111	**297**	0	72.8

Table 11: **What happens to a reward-conflicted contract clause when an optimizer is allowed to touch it.** Every candidate prompt the search evaluated on SST-5 (`gemma-3-27b-it`), classified by the fate of the POLICY-1 clause; counts are candidates pooled across three paired seeds. *Verbatim*: the 284-character clause is found unchanged in the rendered prompt, after normalizing whitespace and JSON over-escaping. *Reworded*: the string POLICY-1 still appears but its text has changed. *Deleted*: no trace of the clause. *Mod. rate* = (Reworded + Deleted) / Candidates, in percent. Detection uses a regular expression and a parser rather than an LLM judge, so the counts are deterministic. A1's zero is definitional (the search space contains no variant of an immutable block) and is reported to anchor the three editable arms rather than as a result in itself. The informative contrast is A4 against A2: a "do not edit" *instruction* eliminates deletion entirely (0 against 223) and raises verbatim retention about fifteenfold (27.2% against 1.8%), yet still leaves 72.8% of candidates with the clause paraphrased.

Arm	Val acc. (proxy)	Test acc. (contract-aware)	Escalation compliance
A1 SPLICE (contract frozen)	49.3 ± 2.9	$\mathbf{58.5 \pm 4.1}$	$\mathbf{66.8 \pm 6.2}$
A2 all-mutable	$\mathbf{59.8 \pm 4.5}$	41.4 ± 0.3	0.0
A3 monolithic	59.4 ± 3.6	41.6 ± 0.4	0.1 ± 0.1
A4 soft-instructed	58.9 ± 4.8	40.2 ± 0.8	1.4 ± 1.6

Table 12: **The ordering inverts between the metric the optimizer sees and the one a deployment pays.** SST-5, `gemma-3-27b-it`, mean $\pm$ s.d. over three paired seeds. The proxy credits sentiment labels only, so a correct escalation scores zero and obeying POLICY-1 forfeits reward; the contract-aware metric credits the escalation instead. Every editable arm beats SPLICE by roughly 10 pp on the proxy and loses by roughly 17 pp on the objective a practitioner deploys against: freezing costs 10.5 pp of proxy ($59.8 \to 49.3$) and buys 17.1 pp of contract-aware accuracy ($41.4 \to 58.5$). The arms are equally good *classifiers*: untriggered label accuracy backs out to $\approx$55.5% (A1) against $\approx$56.2% (A2), so the whole gap is escalation behavior rather than sentiment quality. A2's compliance is exactly 0.0 because every seed deleted the clause, which is a different failure from A1's 66.8%: see Section 5 on text versus behavior.

behavior rather than sentiment quality; the constraint costs nothing on the task itself. *Second, the edits were not cosmetic.* A3 reworded the clause in 316 of 372 candidates and still escalated only 0.1% of triggered snippets: present in name, stripped of force. This is the silent failure a validator cannot catch, and it is why we measure behavior on held-out test rather than stopping at a text-identity check. *Third, A4 is the most informative arm.* "Do not edit" worked in part: zero deletions against A2's 223, and the highest verbatim retention of any editable arm (27.2% against 1.8%). Told not to remove the clause, the optimizer paraphrased it into ineffectiveness instead, and none of that partial protection survived to deployment (1.4% compliance). An instruction shifts the *distribution* of tampering; it does not prevent tampering, and it cannot survive a selection rule that rewards violation. *Fourth, the other three contract classes were untouched by every arm in every candidate.* Optimizers do not vandalize indiscriminately; they trade away precisely the clauses that cost score. That is the argument for freezing by construction rather than auditing after the fact, because a practitioner cannot know in advance which clauses those will be.

ON THE PRICE OF FREEZING. POLICY-1 puts $\approx$26 pp of proxy metric on the table, which makes this the adversarial case for Proposition 2 rather than the typical one: we froze the single clause the optimizer most wanted to remove. The measured price of doing so is 10.5 pp. The bound itself is not testable here: its right-hand side combines γ, which we cannot estimate per benchmark, with a maximum over the counterfactual variant sets $\tilde{\mathcal{V}}_i$, which are unobservable by construction (Remark 15). This arm instead tests the proposition's *shape* (that the price tracks the marginal value of what was frozen) at both ends of that range. The claim of Section E.6, that freezing is nearly free when frozen blocks carry small marginal gains, concerns metric-neutral contracts, and the tool

Strategy	W	D	Best search score*	Step of best
`div3`	4	3	**0.963**	4
`div2`	4	2	0.921	4
`div1`	4	1	0.921	4
`div0`	4	**0**	**0.213**	12
`elitist_fullbatch`‡	6	1	0.926	3

Table 13: **Removing the diversity slot collapses the search on BBH.** The four `div*` rows share beam width $W{=}4$ and differ *only* in the number of reserved diversity slots D, which makes this the clean control for Theorem 1. Without diversity the run is near 0.21 for all 13 steps (every beam member converges to a near-identical score and the search never escapes a single mode), while every $D \geq 1$ variant reaches above 0.92 by step 4. This is the largest single-component effect we measure anywhere in the paper. ‡`elitist_fullbatch` is *not* a diversity control: it differs from the `div*` rows on two axes ($W{=}6$, $D{=}1$) and is included only as a wider-beam reference point; because two axes differ it cannot show whether width substitutes for diversity. *Best score on the step's training batch (216 examples): it measures escape from a mode, not test accuracy. The diversity mechanism that ran is the distinct-parent slot with backfill (Section 2). Every row is a single seed (`random_seed`=42), so the collapse is unreplicated, and the same knob has small or no effect on ARC, GPQA and HelpSteer2 (Section 4). The claim we make is conditional: diversity is insurance against mode collapse and it pays where collapse is a live risk.

schema, safety text and output envelope, untouched by every arm in every candidate, are the evidence for it at zero cost. POLICY-1 is the deliberate stress case at the other end, and 10.5 pp against 26 pp of forgone proxy metric is the number a practitioner should budget for when a frozen clause is expensive.

LIMITS. Three seeds, one task, one target model, and the arms consumed different budgets (456 candidates for A1 against 336, 372 and 408 for A2–A4), which if anything favors A1, so the editable arms' lead on the proxy is not a budget effect. Text preservation is not behavioral compliance, and the gap between them is the sharpest scope condition the experiment produces. With the clause present and no optimization (the seed, A0), the target complies on 96.5% of triggered test examples; in A1 the clause is byte-identical and compliance is 66.8%. The only difference between those two conditions is the optimizer's edits to the *mutable* blocks, which share a token stream with the frozen one and were optimized under a metric that pays for violation. So the most natural reading is that optimized neighbors partly override a frozen clause: freezing protects a block's text, not its effect. Baseline non-compliance does not explain the drop, since A0 is that baseline on the same test split; what the experiment does not identify is which edits to the neighboring blocks erode the clause. The editable arms' 0–1.4% has a different cause in each arm. A2 deleted the clause in most candidates, so there compliance was never possible; A3 and A4 kept it present (A4 retained it verbatim in 111 of 408 candidates and deleted it in none) and complied anyway at $\leq 1.4\%$, so a clause that survives in place but is paraphrased, or surrounded by optimized text, can lose its force entirely. Finally, the mechanism is aimed at silent failures; a contract whose violations raise a parser or schema error is caught by output validation and does not need it.

F.3 Knobs, and Where They Fail

The two axes here are ones a practitioner has to set, and neither of them behaves like a monotone quality dial. Search budget is non-monotonic in N, and trading beam width for per-parent fan-out at a fixed budget did not move the leader; momentum has no measurable effect at its default and a measurable cost at a longer window.

F.3.1 Search Budget: Proposer Draws and Beam Width

We characterize the weakest iterative regime first: Best-of-N sampling with no feedback.

Figure 3 reports BoN accuracy across proposer scales and $N \in \{1, 3, 5, 10, 15\}$ on SST-5 and GPQA Diamond, from a separate run that varies the proposer. Two facts stand out. First, one-shot accuracy scales strongly with proposer capacity: on SST-5 the fitted accuracy rises from $\sim$45% for the smallest proposers to $\sim$55% for the largest. Second, the N-axis is not monotonic: on SST-5 the five N-curves lie within a narrow band at every scale; on GPQA aggregate accuracy peaks at $N{=}10$ and drops at $N{=}15$, as does the small tier; the large tier peaks at $N{=}5$, while the frontier tier rises from

	Arm means (3 seeds)				Paired
Arm	**Val acc.**	**Test acc.**	**Steps**	**Prop. win**	**Δ val**
off	59.44 ± 2.69	56.17 ± 1.76	11.0	0.266	$+0.44 \pm 0.19$
on, w=2	59.44 ± 4.17	55.28 ± 0.77	10.7	0.225	$+0.44 \pm 1.71$
on, w=5 (control)	$\mathbf{59.89 \pm 2.50}$	$\mathbf{56.33 \pm 0.59}$	12.7	0.200	—
on, w=10	57.44 ± 3.56	55.84 ± 1.39	**5.3**	**0.308**	$\mathbf{+2.44 \pm 1.07}$

Table 14: **Momentum: a null at the default window and a real cost at a long one.** SST-5, `gemma-3-27b-it`, three seeds (42/43/44) paired across arms so every contrast is seed-matched rather than confounded by draw; each arm differs from the control in exactly one field. *Steps* is the mean number of steps before early stopping; *Prop. win* is the fraction of proposed children beating their parent. The last column is the paired per-seed difference (control − arm, validation pp; positive means the control is better), which is the meaningful comparison because pairing removes the cross-seed draw variance that dominates the ± on the arm means. Turning momentum off costs +0.44 pp paired with the sign consistent on all three seeds, but that is an order of magnitude below the 2.5 pp cross-seed standard deviation, and on a 300-example split 0.33 pp is a single example, so the defensible statement is *no measurable effect*, not *no effect*. The w=10 arm is different: it loses 2.44 pp paired, consistently on every seed, and terminates at 5.3 steps against the control's 12.7, less than half the search. Its proposal win rate is the *highest* of any arm, which is coherent rather than noisy: a long critique history over-constrains the proposer, so the edits it does produce beat their parent more often but explore a narrower region, and the run early-stops at a worse leader. Momentum's window is a real knob with a wrong end: w=5 is defensible, extending it is not.

N=10 to N=15 (Figure 16 in Section F; its cells carry roughly ±20 pp of sampling uncertainty on a difference). Drawing more samples from the same seed without feedback is not a reliable substitute for capacity or for iteration. The companion axis is beam width W. At a fixed per-step variant budget $Wk = 12$, trading width for per-parent depth (W=4, k=3 against W=2, k=6) did not move the leader's mean. These are single runs per configuration, so we make no claim about the effect on run-to-run variance, and the comparison does not bear on Theorem 2, which concerns the re-evaluation regime SPLICE does not use. It does not test Conjecture 1 either, which varies W at fixed k; that conjecture currently has no empirical handle in this paper. The beam widths used per benchmark ($W = 4$ or 5, Section F.1) are not tuned defaults.

F.3.2 Momentum

Momentum is part of the critique axis of Section 2, and its ablation returns a null. Table 14 isolates it on SST-5 across four arms (off, and window $w \in \{2, 5, 10\}$) with three seeds paired across arms and each arm differing from the control in exactly one field.

Two findings, pointing in different directions. *Turning momentum off costs nothing measurable.* The paired mean is +0.44 pp in the control's favor and the sign is consistent on all three seeds, so we do not claim momentum helps at the default window. But the magnitude is an order of magnitude below the 2.5 pp cross-seed standard deviation, and on a 300-example validation split 0.33 pp is a single example, which two of the three per-seed gaps are. Windows 2 and 5 are likewise indistinguishable. The defensible statement is *no measurable effect*, not *no effect*: three seeds on this benchmark cannot resolve an effect this small. *A long window is measurably worse.* At w=10 the arm loses 2.44 pp paired, consistently on every seed, and terminates at 5.3 steps against the control's 12.7 (less than half the search) while recording the *highest* proposal win rate of any arm (0.308). That combination is coherent rather than noisy: a long critique history over-constrains the proposer, so the edits it does produce beat their parent more often but explore a narrower region, and the run early-stops at a worse leader. Momentum's window is a real knob with a wrong end.

The cross-benchmark evidence agrees. On HelpSteer2, eight matched pairs drawn from a 114-run sweep give a mean effect of +0.009 Spearman ρ with momentum winning three and losing five; on ARC, eight matched pairs give −0.001, nominally hurting six of eight. Across three benchmarks, two metrics and 19 matched pairs there is no effect outside noise on any of them. Two observations belong with that null rather than against it: ARC is saturated (in this sweep the median initial prompt scores 0.963, median run improving 0.004), so it cannot resolve small effects, and the ARC and HelpSteer2 sweeps lack replicates, so a true effect below the noise floor would be invisible.

		GPQA-Diamond, by target tier		
	SST-5	**Frontier**	**Large**	**Small**
step 2	0.92	0.83, 0.58	0.25, 0.75	0.33, 0.08
step 4–5	0.08–0.17	0.17, 0	0.08, 0.08	0, 0
tier mean (step 2)	—	**0.71**	**0.50**	**0.21**
p^{adv} proxy at tier mean, k=3	—	0.98	0.88	0.51

Table 15: **The improving-proposal rate $\hat{p}^\star$, measured, at $Wk = 12$.** Every run logs the per-step fraction of proposed children whose validation score exceeds their parent's; this table is extracted from existing logs, with two folds per GPQA tier. On GPQA the proposer is fixed and the columns are *target* tiers. *Down the column*, coverage is abundant while the prompt still has deficiencies the critique can name, then decays by an order of magnitude as the leader climbs, the state-dependence that the ε-sublevel form of Assumption 1 allows. *Across the row*, $\hat{p}^\star$ increases with target capacity. The last row substitutes the tier mean into the leader-advance rate of Theorem 4, $1-(1-\hat{p}^\star)^k$, as a proxy: the theorem's rate uses only the leader's own k draws, and $\tilde{p}^\star$ counts only gains of at least $\Delta_{\min}$. It orders the tiers the way iteration's advantage over Best-of-N does; the per-step rate $1-(1-\hat{p}^\star)^{Wk}$ is at least 0.94 on every tier and does not. The rate averages over the states the search visited, so it upper-bounds the worst-case quantity in Assumption 1. One improving child out of twelve is already $\hat{p}^\star = 0.08$, so we read only *sustained* zeros as a plateau. Single seed (`random_seed`=42), one run per (fold, strategy).

The implementation tags each momentum entry HELPED, HURT or FLAT by the leader's gain Δ_s relative to $\Delta_{\min}$, and warns when three consecutive significant gains alternate in sign. Under elitist selection with cached scores the leader never falls (Lemma 1), so $\Delta_s \geq 0$: only HELPED and FLAT can occur, and the oscillation warning cannot fire. The absence of oscillation is therefore a property of the selector, not an observation, and momentum acts as a record of which directions moved the leader. Its benefit is undemonstrated rather than refuted, and w=5 should be read as a defensible default rather than a tuned one.

F.4 When Iteration Pays: Capacity, $\hat{p}^\star$, and Cost

The three measurements in this subsection are one argument. Iterative search buys its advantage from the proposer's ability to generate improving edits; that ability is directly measurable; and whether the advantage is worth paying for depends on what the search costs. Taken together they answer the question a practitioner actually asks before adopting the method: *on my task and my model, should I iterate, or sample once and take the best?* They answer it with a diagnostic that can be run in the first few steps rather than with a benchmark table.

The GPQA Diamond sweep varies the target across six models in three tiers while holding the proposer fixed. Eight strategies (Greedy, Beam, Elitist Beam, and Best-of-N at $N \in \{1, 3, 5, 10, 15\}$) run against the targets S1–F2 of Section 4, with proposal temperature 0.9, critique temperature 0.0, three proposals per candidate, patience 5 and up to 50 steps. Validation is 40 questions scored by exact match, one run per cell, and we call each target a *fold*. Averaged over targets, Elitist Beam (64.6%), BoN N=10 (63.3%) and Beam (62.9%) are within noise of one another: one question is 2.5 pp, and a difference between two cells carries about $\pm$20 pp of sampling uncertainty. The per-tier view (Figure 15 in Section F) is where a pattern shows: iterative strategies score highest on the frontier tier, and no strategy consistently wins on the small tier, where Section F.4.1 measures $\hat{p}^\star = 0.21$, so $1-(1-\hat{p}^\star)^k \approx 0.51$ at $k = 3$. Section F.7 gives the BoN-scaling, train–validation and cost views of the same sweep.

F.4.1 The Improving-Proposal Rate $\hat{p}^\star$

THE MEDIATING QUANTITY, MEASURED. Table 15 reports $\hat{p}^\star$, the per-step fraction of proposed children whose validation score exceeds their parent's. This is the success event of Assumption 1, averaged over the parents the search visited, so it estimates the improving-proposal probability of that assumption on average rather than in the worst case. It is the single most informative diagnostic in the paper, because it reads in two directions at once and each direction discharges a different claim.

Down the column it traces state-dependence. Coverage is abundant while the prompt still has deficiencies the critique can name (0.92 at step 2 on SST-5, 0.58–0.83 on frontier GPQA folds), then decays by an order of magnitude as the leader climbs, reaching zero in several late steps. This is an empirical trace of $\hat{p}^\star(\varepsilon)$ as ε shrinks, and it is what the ε-sublevel form of Assumption 1 predicts rather than something in tension with it. The terminal zeros are the stopping condition being met, not the assumption failing: they occur after the leader has plateaued, at states whose remaining gap is below what the search can resolve, which is why SPLICE exits on consecutive no-improvement steps rather than spending the remaining budget.

Across the row it gives a measured mechanism for the target-capacity effect of Section 4. With the proposer fixed, tier means at the step 2 are 0.71 (frontier), 0.50 (large) and 0.21 (small): $\hat{p}^\star$ *increases* with target capacity, so the improving-proposal rate is a property of the (proposer, target, task) triple, not of the proposer alone. Which rate to read it through matters. Substituting $\hat{p}^\star$ into the per-step reachability rate gives $1-(1-\hat{p}^\star)^{Wk} \geq 0.94$ for all three tiers at $Wk = 12$, which predicts that iteration should pay everywhere, and it does not on small targets. Substituting it into the leader-advance rate of Theorem 4 gives $1-(1-\hat{p}^\star)^k = 0.98$, 0.88 and 0.51 at $k = 3$, an ordering that matches where iteration separates from one-shot sampling. The second substitution is a proxy: the theorem's rate p^{adv} uses only the leader's own k draws, and $\tilde{p}^\star$ (Assumption 5) counts only gains of at least $\Delta_{\min}$, while $\hat{p}^\star$ counts every parent's children and any positive gain. We did not compute the leader-only, thresholded rate, so these values indicate the advance rate rather than measure the theorem's quantity. The convergence-relevant floor is therefore $1/k \approx 0.33$, not the per-step resolution floor $1/Wk \approx 0.08$, and the small-tier mean sits below it; small-tier runs also stop early, after about six steps.

Two consequences we would rather state than leave implicit. First, this makes the scope condition of Section 4 operational: SPLICE's iterative advantage needs a $\hat{p}^\star$ well above $1/k$, and below that one-shot sampling is a reasonable choice when the task is saturated, though not always, since on the judge task (Section F.5) Best-of-N regresses below its seed. Second, because the rate is observable within the first few steps of any run, it doubles as a cheap pre-flight check on a practitioner's own (task, proposer, target) triple. The caveats are in Table 15: these are single-seed runs, one per (fold, strategy); the observed rate averages over the states the search visited, so it upper-bounds the worst-case $p^\star(\varepsilon)$ of Assumption 1 rather than estimating it; and it cannot distinguish convergence to ε-optimality from a radius-one local maximum, since both appear as zeros.

F.4.2 Cost

Cost splits into two questions that are easy to conflate, and only one of them admits a fair comparison against prior work.

INFERENCE-TIME COST OF THE RETURNED PROMPT. This is measured and comparable, since every method is evaluated on the same BBH test set through the same task model. Table 16 reports it: SPLICE's optimized prompt costs $1{,}452$ total tokens per example against GEPA's $1{,}727$, a $1.19\times$ saving driven almost entirely by $2.9\times$ fewer completion tokens (128 against 373) at near-identical prompt length ($1{,}325$ against $1{,}354$). The claim we make is therefore *comparable prompt size, substantially shorter outputs* rather than a large total-cost win, and two caveats travel with it: the completion figure is low enough relative to every other method that it may in part reflect suppressed intermediate reasoning, and the harness evaluates SPLICE through a raw completion call while the other rows pass through a chain-of-thought wrapper, an asymmetry that inflates their completion counts.

OPTIMIZATION-TIME COST WITHIN SPLICE. Across selectors on GPQA, where the instrumentation is identical for every row, Elitist Beam reaches the highest validation accuracy on 24% fewer tracked tokens than Beam (1.06M against 1.39M per fold, excluding candidate scoring; Table 17 in Section F), a 26% better cost-per-accuracy ratio. On BBH the diversity arms reach their best score at step 4 while `div0` never passes 0.213 in 13 steps (Table 13), so the diversity arms also reach their best much earlier; we do not have per-arm token counts for that sweep and so give no ratio.

On GPQA, Elitist Beam costs 12–$41\times$ a Best-of-N run per fold depending on N (Table 17), and BoN(N=10) lands 1.3 pp below it at roughly 6% of the token cost, a difference this sample cannot resolve. So our cost argument is *elitist-versus-beam*, not *iterative-versus-one-shot*: where budget binds on a saturated task, one-shot sampling is a reasonable choice (though not on the judge task,

Method	Prompt tok/ex	Completion tok/ex	Total tok/ex
Baseline (no optimization)	365	350	714
GEPA	1,354	373	1,727
ContraPrompt	1,021	348	1,368
SPLICE	1,325	**128**	**1,452**

Table 16: **Inference-time cost of the returned prompt.** BBH, 405 test examples, identical task model (`claude-haiku-4-5`) for every row. Each column is a rounded per-example mean, so a total can differ from the sum of its parts by one. SPLICE's optimized prompt costs $1.19\times$ less per example than GEPA's, and the gain is almost entirely on the output side: $2.9\times$ fewer completion tokens at near-identical prompt length (1,325 against 1,354). The defensible claim is therefore *comparable prompt size, substantially shorter outputs*, not a large total-cost win. We report per-example figures only and omit totals, because only 181 of 405 SPLICE calls survive in the response cache, so its totals are not full-test-set totals and would not be comparable with the other rows'. Two caveats travel with the completion-token figure: it is low enough relative to every other method that it could in part reflect suppressed intermediate reasoning rather than pure efficiency, and the harness evaluates SPLICE through a raw completion call while the other rows pass through a chain-of-thought wrapper, an asymmetry that inflates their completion counts. *Optimization-time* cost is deliberately not compared here; see Section F.4.2.

where it regresses below the seed), and the $\hat{p}^\star$ diagnostic of Section F.4.1 is the cheap way to decide which regime a given (task, proposer, target) triple is in.

WHAT WE DO NOT REPORT, AND WHY. There is no SPLICE-versus-GEPA comparison on optimization-time cost. Our own instrumentation does not populate the candidate-scoring bucket, which is the dominant term (each scored candidate consumes a full evaluation batch, so the tracked figures understate true optimization cost by roughly $40\times$), and the GEPA reproduction logged no optimizer tokens at all, its reflection and mutation calls never having been cached. Publishing a partially instrumented number of ours against an absent number of theirs, in our own favor, is precisely the asymmetry a cost table should not contain. Closing this requires instrumenting both sides and re-running under a matched optimizer budget, which we scope as future work rather than estimate here.

F.5 Application: LLM-as-Judge Rubric Calibration

The benchmarks above all optimize prompts whose output is the system's primary task response. Here we apply SPLICE to a task of a different kind: optimizing the rubric prompt given to an LLM-as-judge so that its ratings align with human annotations. The framework is unchanged (the rubric text is the block-structured state, the judge's output is an intermediate signal, and the scalar metric is the rank correlation of that output with ground truth), but the target has shifted from "get the task right" to "predict human preferences correctly." Judge rubrics are a natural application regime for the method: they are long, multi-section artifacts whose failure modes are mis-calibrations rather than outright errors, and they are increasingly used as the scoring mechanism for other prompt-optimization pipelines.

Task and data. We use the NVIDIA HelpSteer2 dataset [80], which provides human ratings for assistant responses across five criteria on a 1–5 scale: helpfulness/understanding, correctness/completeness, coherence/clarity, language complexity, and succinctness/verbosity. The judge receives a rubric prompt containing an overall-quality rubric and a per-level description for each of the five criteria; given a (user prompt, assistant response) pair, it emits six scalar scores. The metric is the mean Spearman rank correlation between the judge's scores and the human ratings, averaged across the five criteria. Scoring uses a stratified 400-example validation split; each step's training batch has 64 examples. Proposal temperature is 0.9 and critique temperature 0.0.

Judge models. We evaluate across the six judge models S1–F2 of Section 4, two per tier (Small, Large, Frontier). The initial hand-crafted rubric reaches mean Spearman $\rho = 0.555$ across these six models, a competitive baseline that sets a high bar for any automated optimizer.

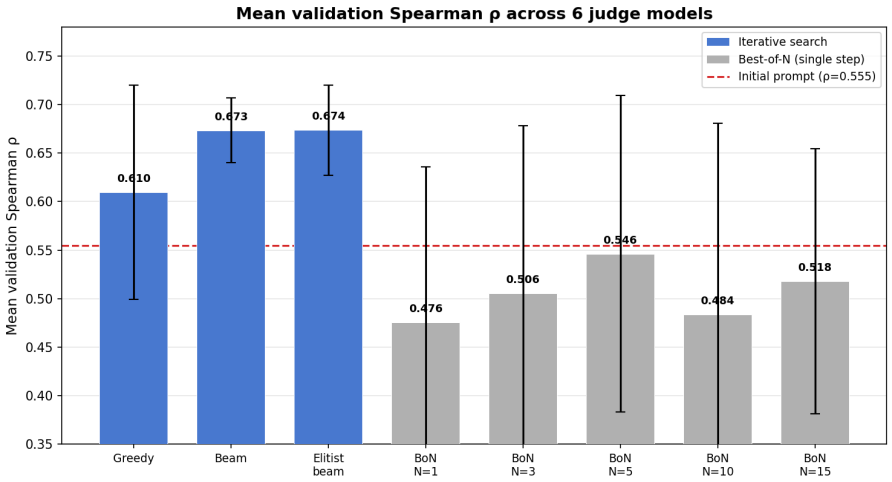


Figure 10: **Judge Alignment on HelpSteer2: iterative search vs. single-step sampling.** Mean validation Spearman ρ between the optimized judge and human annotations across six judge models (error bars: standard deviation across models). The initial hand-crafted rubric (red dashed line, ρ=0.555) is a competitive baseline that none of the Best-of-N configurations matches on average. All three iterative strategies improve over the seed; Best-of-N *regresses below* it.

Results. Figure 10 reports mean validation Spearman ρ across the six judge models. The best iterative strategy (Elitist Beam, D=2, `parent` mode) reaches $\rho = 0.674$, followed by Beam at 0.673 and Greedy at 0.610. The best single-run configuration is F1 with Elitist Beam, at $\rho = 0.736$. None of the Best-of-N configurations match the iterative results: the best BoN mean is 0.546 (N=5), more than 0.12 below the Elitist Beam mean, and BoN mean-over-N is non-monotonic (N=1:0.476, 3:0.506, 5:0.546, 10:0.484, 15:0.518). The gap between iterative search and single-step sampling is wider on this task than on any of the public benchmarks in the main text.

Single-step sampling regresses below the seed. On this dataset Best-of-N sampling *actively degrades* the prompt relative to the hand-crafted seed. The mean improvement over the seed is -0.049 averaged across all five BoN configurations, so on average BoN leaves the prompt worse than the seed. Iterative strategies move in the opposite direction, improving over the seed by $+0.097$ on average. Variance decomposes the effect further: across judge models, the standard deviation of BoN scores is 0.166, compared to 0.063 for iterative strategies. On this task BoN is not merely weaker in expectation; for individual judge models it is sometimes catastrophic (e.g. $\rho = 0.18$ on L2 with BoN N=1, compared to 0.614 for Elitist Beam on the same model). The iterative-vs-BoN gap is therefore not a question of whether more samples would eventually match search: on a task where the prompt is sensitive to small-scale calibration choices, up to N=15, proposer-only sampling does not recover what iterative critique + selection delivers.

The elitist property prevents leader regression. Elitist Beam (0.674) and Beam (0.673) achieve nearly identical means, but the mechanism differs in a way that matters for worst-case behavior. Under Beam, the final leader may be worse than the best candidate seen during optimization; under Elitist Beam with the cached-scoring convention of Section 2, the leader score is monotone non-decreasing (Lemma 1). Greedy ($\rho = 0.610$) suffers most from regression: on S2 its final leader drops to 0.402 even though a stronger candidate appeared earlier in the search. Across the six judge models, greedy's standard deviation is 0.110, approaching BoN-level variance despite iterative search, a direct consequence of maintaining only a single candidate with no fallback when it regresses. Beam and Elitist Beam standard deviations across the six judge models are 0.033 and 0.046 respectively, against 0.166 for BoN.

What SPLICE discovered. Inspecting the optimized rubric reveals four calibration strategies that the seed lacks, each addressing a known failure mode of LLM judges. *(i) Descriptive vs. evaluative dimension separation.* Language complexity and verbosity are descriptive (a simple response is not worse than a complex one), whereas helpfulness, correctness, and coherence are evaluative. The

optimized rubric marks the first two as "descriptive only" and instructs the judge that low scores on these dimensions must not reduce the overall score, a distinction absent from the symmetric seed. *(ii) Anti-deflation gates.* The optimized prompt contains explicit "default high" clauses and mandatory upgrade-before-downgrade tests ("before assigning score 3, ask: can I name ONE specific factual claim that is clearly wrong AND central to the answer? If not, you MUST score 4") that counteract the well-documented tendency of LLM judges to under-score. *(iii) Structured scoring decision trees.* Rather than the seed's per-level prose, the optimized rubric encodes a four-step decision procedure (topic relevance $\rightarrow$ task understanding $\rightarrow$ substantive correctness $\rightarrow$ named-flaw test), where each step maps to a minimum score. *(iv) Dual-track overall scoring.* Two independent tracks (core-quality threshold on helpfulness, correctness, and coherence; or arithmetic mean above a cutoff) each suffice for the top overall score, preventing descriptive dimensions from dragging down the aggregate. These mechanisms are specific, interpretable, and align with known pathologies documented in the LLM-as-judge literature; SPLICE discovers them from rank-correlation feedback alone, without any auxiliary supervision on where the judge is mis-calibrated.

This appendix provides supporting diagnostics for the experiments of Section 4. The head-to-head benchmark numbers, the contract experiment, the component ablations, the capacity and $\hat{p}^\star$ analyses, and the judge-alignment application are all in the main text; what follows is the material that supports those claims without being one of them: the strategy sweep against published prior-method numbers on SST-5 (Section F.6), the search-tree analysis on GPQA Diamond (Section F.7), the remaining GPQA Diamond diagnostics (per-tier bars, BoN scaling, generalization gap and cost–accuracy Pareto; Section F.7) and candidate-lineage and score-progression traces on SST-5 and ARC-Challenge (Section F.8).

F.6 Additional Search Strategy Ablations.

Figure 11 places every SPLICE variant shown (Elitist Beam with $D{=}1$, `parent` and `lineage` modes), the Greedy and Beam arms, and every Best-of-N baseline on the same axes as the published prior-method numbers, for the two SST-5 target models. Three points follow. First, the top two positions on both target models are occupied by SPLICE variants with $D{=}1$ diversity (`parent` and `lineage` modes). The figure has no $D{=}0$ elitist arm, so it shows that the $D{=}1$ variants rank highest, not that the slot is the reason. Second, the best elitist variant leads the best BoN+critique configuration by 3.0 pp on Gemma and 1.8 pp on DeepSeek, while Greedy and Beam fall below it, so on SST-5 the margin over one-shot sampling comes from elitist search, not from iteration alone. Third, critique changes plain BoN by -1.6 to $+2.8$ pp (positive in nine of ten pairs; compare Figure 4a).

F.7 GPQA Diamond: Search-Tree Topology

Figure 13 expands the "Search Strategy" discussion of the main text: the search-tree grid visualizes the exploration footprint of each selector on frontier target models. This figure, Figure 20 and the case studies of Section F.7.5 come from a separate GPQA run with the same strategies and targets as the sweep of Section F.4, so their final scores need not match Figure 15. In the panel titles of the GPQA figures, Claude Sonnet 4 and Claude Opus 4 denote F1 and F2, and the other names denote S1–L2 in the order of Section 4.

Greedy produces $\sim$19-node chains, Beam produces 61–97-node trees, and Elitist Beam sits between at ~ 40 nodes.

The main text (Section F.4) summarizes this sweep; Table 17 gives its aggregate, and this subsection holds the remaining views of the same runs: per-tier bars, Best-of-N scaling, the generalization gap, the cost–accuracy Pareto, search-tree topology across all six models, and three qualitative case studies. For reference, the sweep is eight strategies (Greedy, Beam, Elitist Beam, and Best-of-N at $N \in \{1, 3, 5, 10, 15\}$) across six target models (S1–F2, Section 4). All runs use one fixed proposal agent (temperature 0.9) and one fixed critique agent (temperature 0.0); the validation split has 40 questions, scored by exact match. Iterative strategies use 3 proposals per candidate, early-stopping patience of 5, and up to 50 steps.

Figure 15 isolates each tier. On **frontier** models, Beam (87.5%) and Elitist Beam (83.8%) score highest, with BoN $N{=}3$ close behind (81.3%). On **large** models, Elitist Beam is the single best strategy (65.0%), with BoN $N{=}5$ and $N{=}10$ near 60%. On **small** models the bars span 39–59$\%$

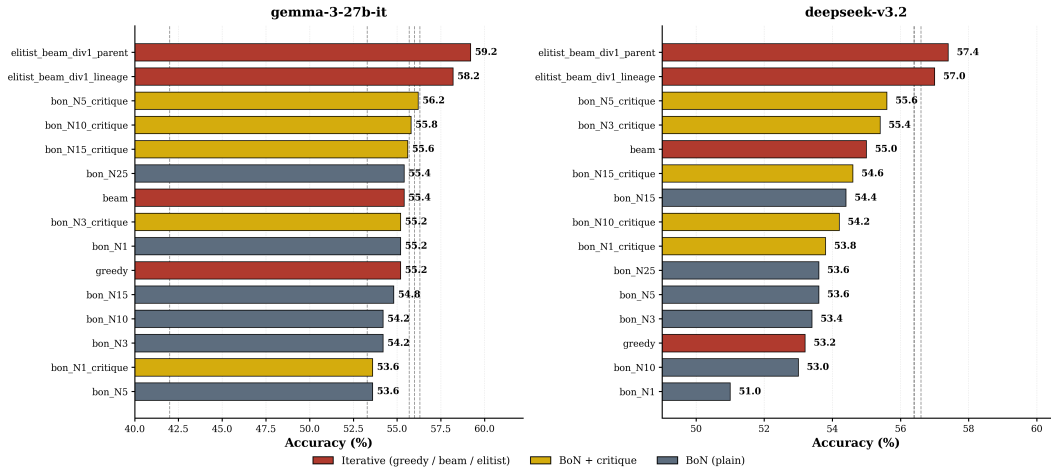


Figure 11: **Only the elitist $D{=}1$ variants beat critique-guided one-shot sampling.** Bars report test accuracy of the final leader, mean over three runs, for every iterative strategy (red: Greedy, Beam, Elitist Beam) and every one-shot baseline (yellow: BoN with critique; grey: plain BoN). Dashed lines mark published numbers for prior optimizers from [89] (left) and [94] (right). The elitist variants lead the best BoN+critique arm by 3.0 pp (Gemma) and 1.8 pp (DeepSeek); Greedy and Beam fall below it.

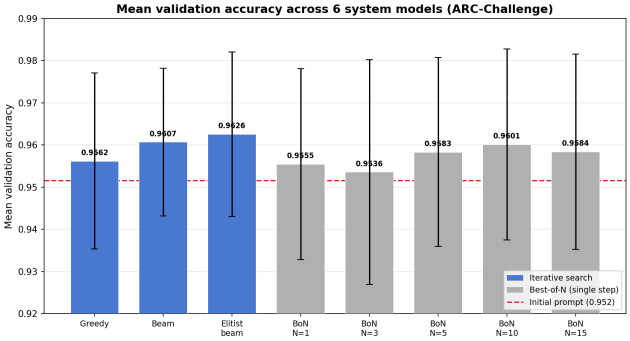


Figure 12: **On ARC-Challenge the strategies compress.** Validation accuracy, mean over the six optimizer models S1–F2; the dashed line is the initial prompt. The SST-5 counterpart is Figure 4b.

with overlapping error bars and BoN $N{=}10$ is highest: strategy choice matters less than model capability at this scale.

F.7.1 Best-of-N scaling is non-monotonic

Figure 16 tracks how BoN accuracy changes as N increases. The key finding is that *scaling is non-monotonic*: the overall and small-tier curves peak at $N{=}10$ and drop at $N{=}15$, the large tier peaks at $N{=}5$, and the frontier tier peaks at $N{=}3$ and rises again from $N{=}10$ to $N{=}15$. Small models gain most from larger N (38.8% to 58.8%). The more candidates are ranked on the same 40 validation questions, the likelier the top one is lucky rather than good; with one run per cell, these differences are also within sampling noise.

F.7.2 Generalization gap: iterative scores are reliable proxies

Figure 17 shows the gap between training and validation accuracy, strategy by strategy. Iterative strategies hover near zero: their training scores, averaged over many batches, track validation. BoN strategies show large *negative* gaps (validation $\gg$ train): one training batch is a noisy score, and the validation score of the selected candidate is biased upward by selection. Because every arm selects on $\mathcal{D}_{\text{valid}}$, the gap does not change which prompt BoN returns.

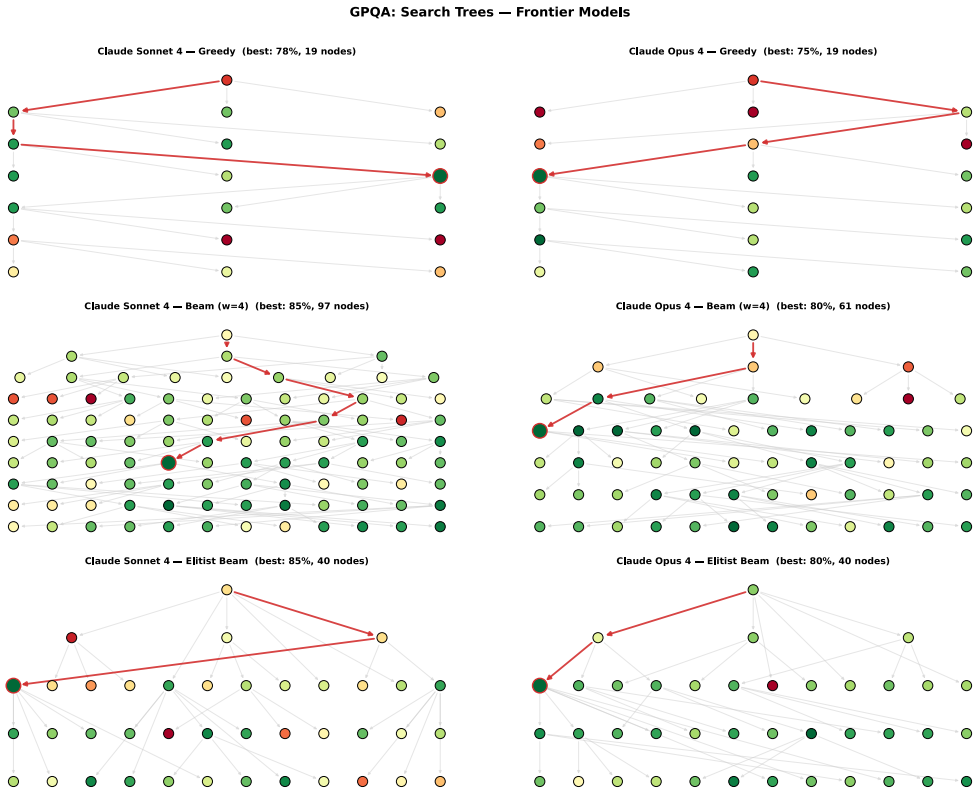


Figure 13: **(Search trees, frontier models.)** Rows: selection strategies (Greedy, Beam $W{=}4$, Elitist Beam); columns: frontier target models. Node color encodes validation score (red = low, green = high); the red path is the best-candidate lineage. Greedy produces narrow $\sim$19-node chains: once it finds a good candidate, it commits. Beam generates bushy 61–97-node trees exploring many parallel lineages. Elitist Beam sits between at $\sim$40 nodes.

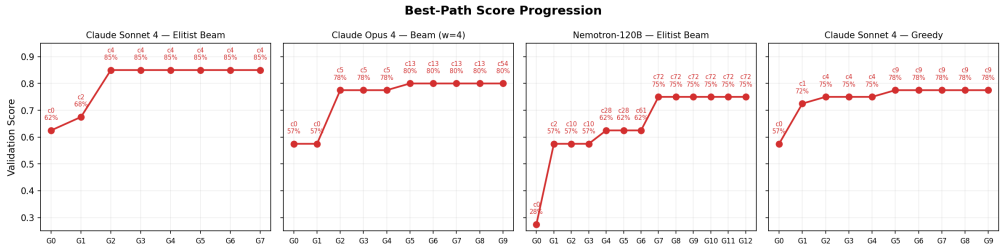


Figure 14: **(Candidate Lineage.)** Validation score along the best-scoring lineage for four GPQA-Diamond runs; x-axis is generation along the lineage. Frontier targets (F1, F2) reach their best score in 3–4 nodes counting the seed; L1 needs 6. The Greedy run (rightmost) improves at generations 1, 2 and 5.

Figure 18 confirms this from a different angle. Iterative runs (circles) cluster along the diagonal: what you see in training is what you get in validation. BoN runs (squares) sit far above the diagonal: their training scores are unreliable. The pattern holds across all model tiers, indicating it is a structural property of single-step versus multi-step optimization rather than a model-specific artifact.

F.7.3 Cost–performance tradeoff

Table 17 gives accuracy and token cost per strategy, averaged over the six targets, and Figure 19 plots token cost against validation accuracy for every (model, strategy) run. The runs split by cost rather than accuracy: BoN runs use 25–90K tokens and iterative runs 250K–2.7M, while both span roughly 30–88% accuracy, ordered mainly by target model. For budget-constrained settings, BoN $N{=}10$ at

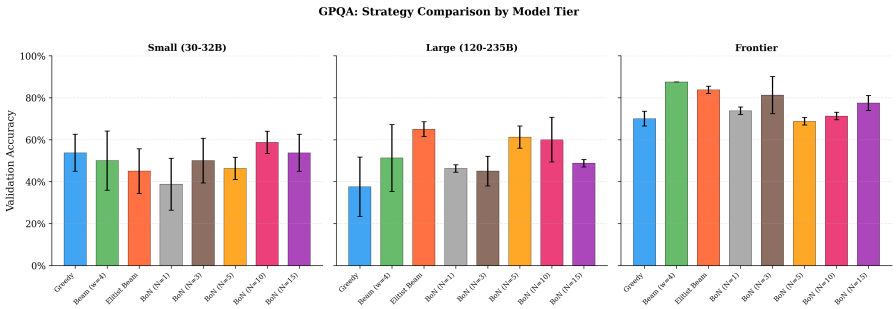


Figure 15: **GPQA strategy comparison by model tier.** Beam and Elitist Beam lead on frontier models and Elitist Beam on large ones; on small models no strategy consistently wins. Tick label "w=4" is the beam width W=4.

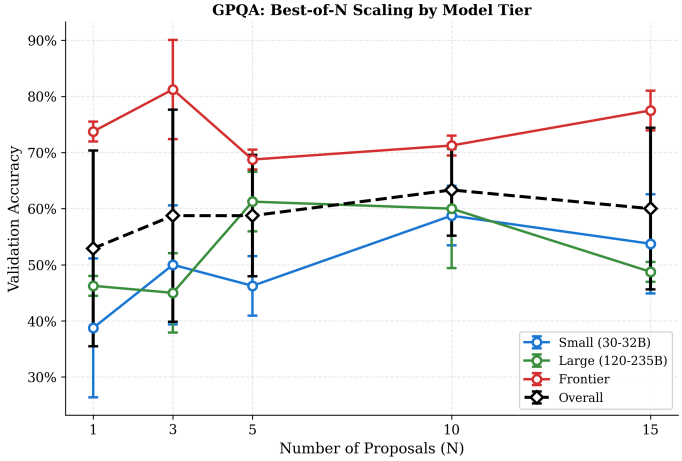


Figure 16: **BoN scaling by tier on GPQA Diamond.** Accuracy is non-monotonic in N; the overall and small-tier curves peak at N=10, and small models gain most.

67K tokens offers 63.3% accuracy, competitive with iterative methods that cost 5–21× more; this is the one regime in which one-shot sampling is a reasonable default, and it relies on a validation split large enough to control selection noise.

F.7.4 Elitist Beam search trees across all six models

Figure 13 shows the search-tree footprint across strategies on frontier targets; the complementary view in Figure 20 fixes the strategy at Elitist Beam and sweeps the target model across all six of our tiers. Frontier models converge in ∼40 nodes, matching the frontier column of Figure 13, while large models need up to 136 nodes, 3.4× more exploration to find a good prompt. This mirrors the per-tier accuracy results: harder optimization landscapes require more search, and the elitist selector supplies it without sacrificing the monotonicity guarantee of Lemma 1.

F.7.5 Case studies: what iterative refinement discovers

Three representative F1 runs illustrate how a prompt is refined along the best-scoring lineage.

Elitist Beam (62% → 85%). The seed prompt (c_0) contains only a bare objective: "Answer the following multiple choice question by reasoning through it step by step." In generation 1, the optimizer adds a `<role>` ("expert scientist and problem solver"), a structured `<approach>` with numbered steps, and a few-shot example, but the score rises only to 68% (c_2). The breakthrough comes in generation 2 (c_4, 85%): the approach is rewritten to enforce *plain prose* reasoning ("no markdown headers, no numbered sub-sections, no bullet lists") and a 2–5 sentence budget, plus the

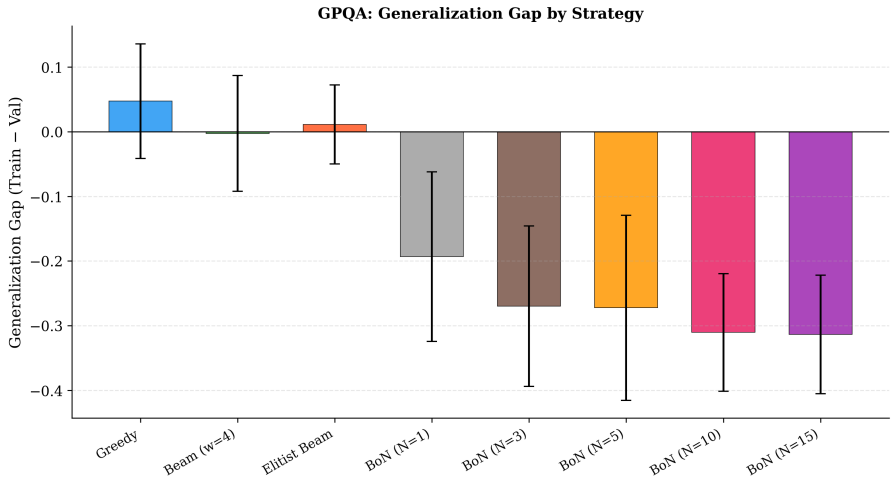


Figure 17: **Generalization gap (Train − Val) by strategy on GPQA Diamond.** Iterative strategies are near zero; BoN strategies have large negative gaps of -19 to -31 percentage points, meaning single-batch training accuracy is not a usable proxy for validation performance. Tick label "$w{=}4$" is the beam width $W{=}4$.

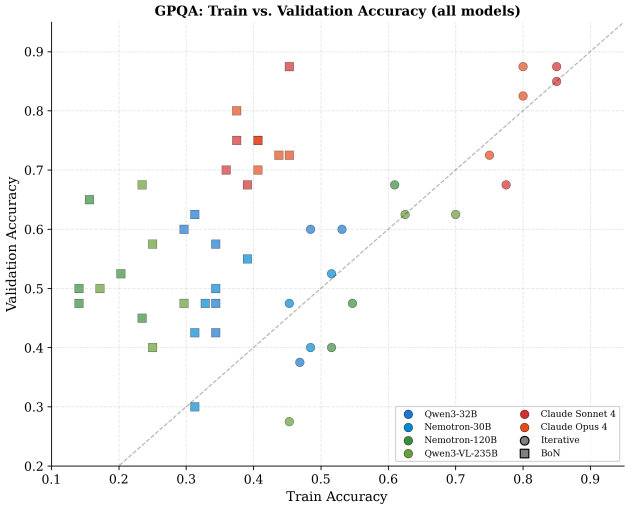


Figure 18: **Train vs. validation accuracy on GPQA Diamond.** Iterative strategies track the diagonal; BoN strategies overshoot.

objective is tightened to require *sufficient* justification. This formatting constraint prevents the model from wasting tokens on verbose structure and forces it to reason concisely, a lesson the optimizer discovered from critique feedback on failed examples.

Greedy (57% → 78%). The Greedy run shows a different refinement trajectory across four nodes. Generation 1 (c_1, 72%) introduces a decisive persona ("prioritize reaching a clear, correct final answer over exhaustive explanation") and a concise approach. Generation 2 (c_4, 75%) adds a strict word budget ("if your reasoning would exceed 8 sentences, stop and state your best answer") and option-scanning heuristics. Generation 5 (c_9, 78%) bans all markdown formatting and imposes a hard 150-word reasoning cap. Each refinement targets a specific failure mode: verbose responses that get truncated before stating an answer, markdown that confuses answer extraction, and over-analysis that leads to second-guessing.

Table 17: **GPQA validation accuracy by search strategy** (Elitist Beam here uses D=0, so it differs from Beam only in keeping parents), mean ± std **across the six target models**, *not* across seeds. Every run is `random_seed`=42, one run per (model, strategy), so the large spreads are substantially the tier effect of Figure 15 rather than run-to-run variance; no seed replicates exist for this sweep. Train–val gap is a diagnostic, not an objective: iterative training scores are averaged over the steps' training batches, while a BoN run logs a single batch; every arm selects on $\mathcal{D}_{\text{valid}}$. Gap is computed from unrounded means and can differ from the rounded columns by 0.1. The sweep has no test split, so the gap says nothing about validation optimism.

Strategy	Models	Val (%)	Train (%)	Gap	Tok. (K)
Greedy (W=1)	6	53.8 ± 16.4	58.5 ± 14.3	+4.7	342
Beam (W=4)	6	62.9 ± 21.3	62.7 ± 16.3	−0.3	1,394
Elitist Beam	6	$\mathbf{64.6 \pm 18.1}$	$\mathbf{65.7 \pm 15.3}$	+1.2	1,060
BoN (N=1)	6	52.9 ± 17.4	33.6 ± 7.4	−19.3	26
BoN (N=3)	6	58.8 ± 18.9	31.8 ± 11.2	−27.0	34
BoN (N=5)	6	58.8 ± 10.8	31.5 ± 9.5	−27.2	44
BoN (N=10)	6	63.3 ± 8.2	32.3 ± 9.1	−31.0	67
BoN (N=15)	6	60.0 ± 14.4	28.7 ± 10.8	−31.4	86

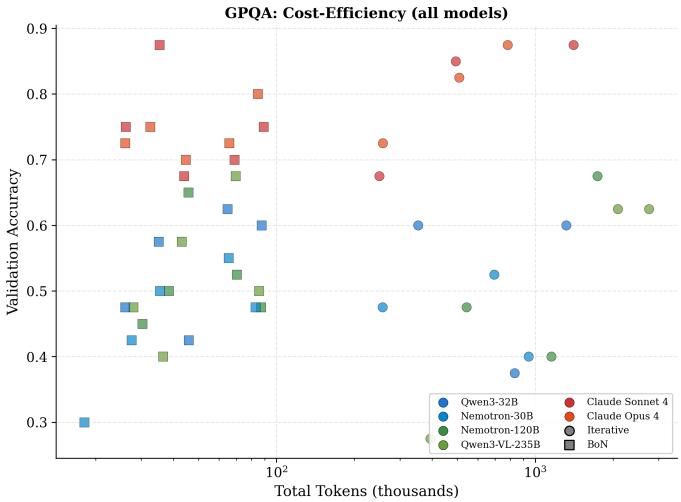


Figure 19: **Cost vs. accuracy on GPQA Diamond.** Circles = iterative, squares = BoN; colors = target model.

Beam W=4 **(52% → 85%).** The Beam run traverses seven nodes ($c_0 \to c_2 \to c_9 \to c_{22} \to c_{33} \to c_{42} \to c_{53}$), the longest best-path lineage among our runs. Generation 1 (c_2, 65%) adds a SCAN/FOCUS/REASON structure and a "no backtracking" directive. The critical insight arrives at generation 5 (c_{22}, 68%): the objective is rewritten to declare that "reasoning without a final answer is a failure regardless of quality," and the approach shifts to answer-first-then-verify. Generation 7 (c_{33}, 70%) escalates further: "state your best answer BEFORE verification." Generation 8 (c_{42}, 78%) introduces explicit word budgets and simplifies the approach to three steps. The final node (c_{53}, 85%) imposes a hard 250-word cap and adds a fourth example. The trajectory reveals a consistent theme: each generation tightens the contract between reasoning depth and answer reliability, progressively learning that *completing the answer matters more than showing all work*.

Recurring refinement themes. Across all three runs three patterns emerge. (i) *Format discipline*: early steps add formatting constraints (plain text, no markdown, word limits) that prevent answer-extraction failures. (ii) *Reasoning calibration*: later steps tune the depth of reasoning, encouraging quick elimination for conceptual questions and systematic derivation for multi-step problems. (iii) *Few-shot accumulation*: examples are added incrementally ($1 \to 2 \to 3 \to 4$ across generations), each targeting a different question archetype (recall, calculation, synthesis, spectral analysis). None of

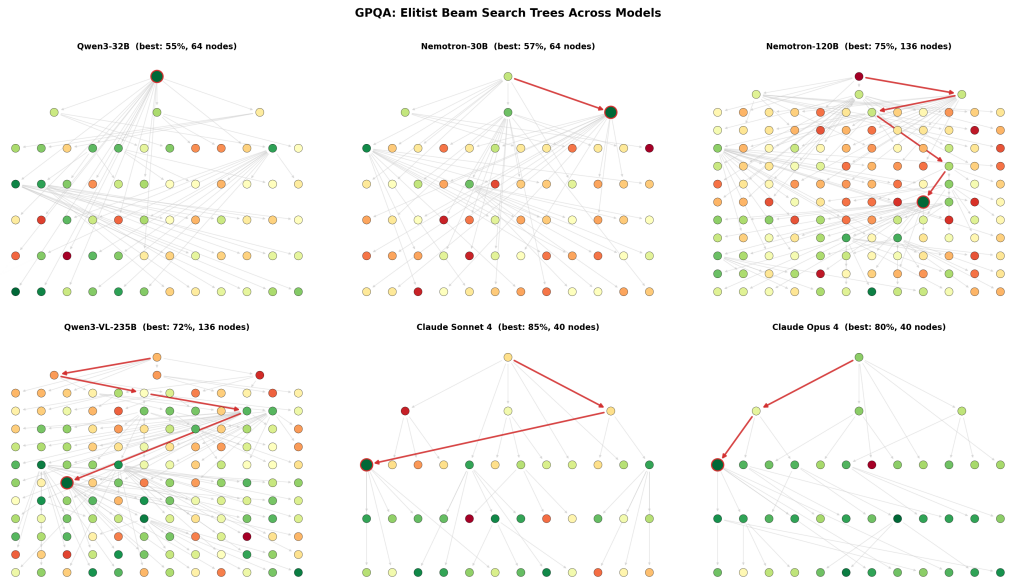


Figure 20: **Elitist Beam search trees across all six target models on GPQA Diamond.** Frontier models (bottom middle and right) converge in ~ 40 nodes, small models (top left and middle) take 64, and large models (top right, bottom left) need up to 136. Node color encodes validation score (red = low, green = high); the red path traces the lineage of the best candidate.

these themes are present in the seed; all are discovered from scalar validation feedback and textual gradients alone.

F.8 Candidate Lineage and Score Progression on SST-5

Figure 13 in Section F.7 shows the search-tree footprint on GPQA Diamond. The analogous view for SST-5 is reported here.

F.8.1 SST-5

Figure 21 traces the best-leader validation accuracy along the search step index, and Figure 22 shows the candidate-lineage graph on the best SST-5 run. The same two patterns observed on GPQA recur. *(i) Monotone best-path score.* The leader accuracy is non-decreasing along the best-scoring lineage, the empirical signature of Lemma 1 under the cached-scoring convention. *(ii) Early but not only early gains.* About two-thirds of the 5.2 pp gain arrives by generation 3, and the last $+1.4$ pp at about generation 9. This matches the improving-proposal rate being highest early in the search (Section F.4.1).

Beam diversity and proposer success rate. Two further per-step diagnostics on SST-5 track how the selector and proposer behave inside the search loop. Figure 23 reports the best validation score and the spread of beam scores at each generation. After step 1 the spread stays small, so the beam's members score alike while the leader climbs.

Figure 24 reports the complementary quantity on the proposer side: the fraction of candidate children whose validation score exceeds their parent's, measured per search step. Under the deterministic monotonicity of Lemma 1 the *leader* is non-decreasing by construction, but the per-proposal success rate is an empirical quantity: it says how often a textual-gradient-conditioned proposal actually improves on its parent. That fraction is itself a per-proposal rate (it estimates $\hat{p}^{\star}$ directly, not the per-step reachability probability), and on SST-5 it is $11/12$ at step 2, at most $3/12$ from step 3, and zero in most steps after step 6, falling as the leader climbs. This run is not the one tabulated in Table 15, whose SST-5 column falls to 0.08–0.17 by steps 4–5; the two should be read as different runs, not as one rate.

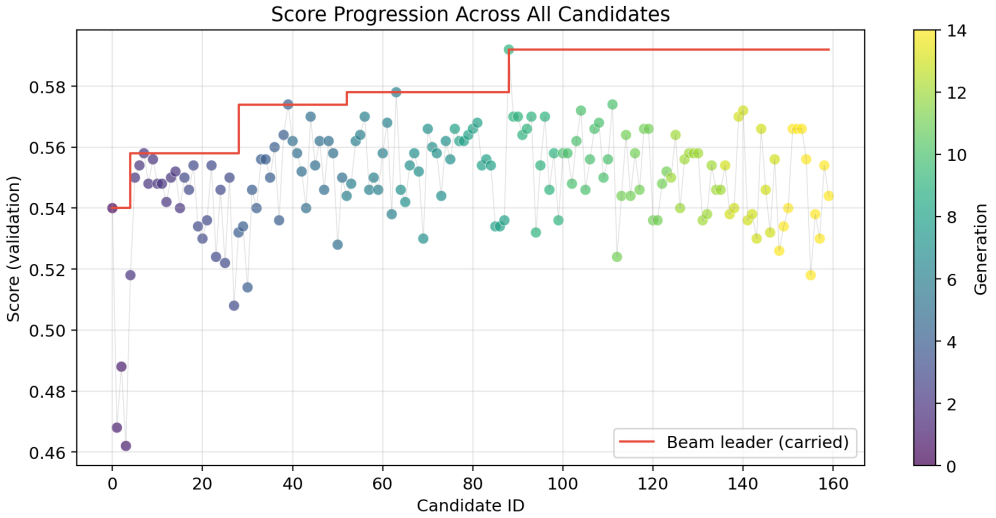


Figure 21: **SST-5: the carried leader never falls, and most of its gain comes early.** One run. Dots are candidates by ID, colored by generation; the red line is the beam leader, which rises from 54.0% to 59.2%, about two-thirds of the gain by generation 3.

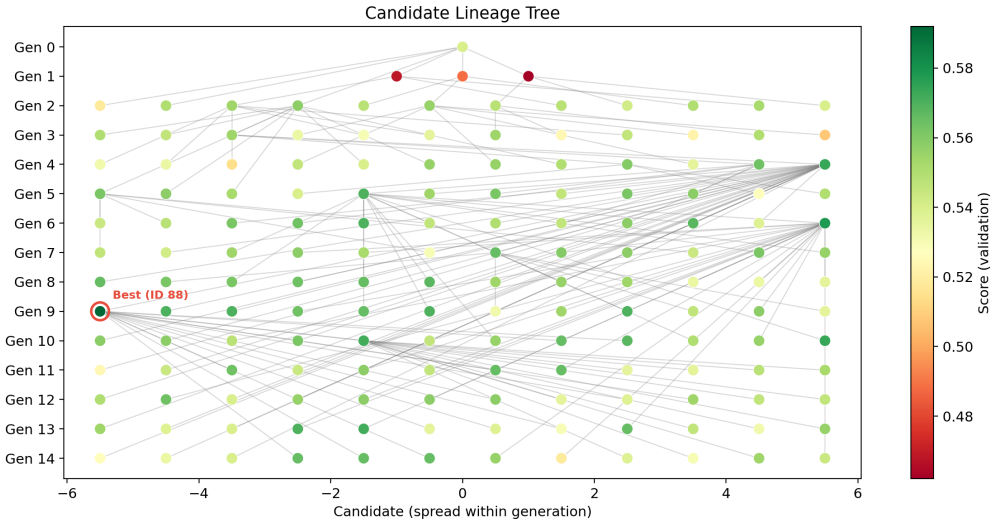


Figure 22: **SST-5: the search fans out early, then concentrates on one branch.** Candidate lineage graph for one run. Nodes are candidate prompts colored by validation accuracy (red = low, green = high); edges connect a candidate to the parent(s) it was proposed from. The ringed node is the final best candidate. The fan-out pattern at early generations reflects the Elitist Beam selector filling its diversity slots (Theorem 6) before converging on the best-scoring branch.

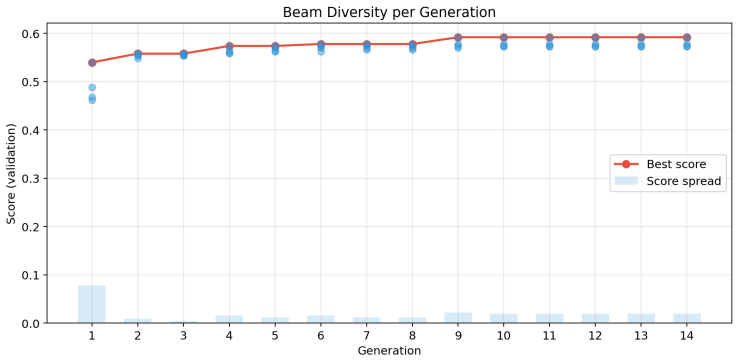


Figure 23: **SST-5: after the first step, beam members score alike.** Best validation score (line) and the spread of beam members' scores (bars) at each generation of the Elitist Beam run.

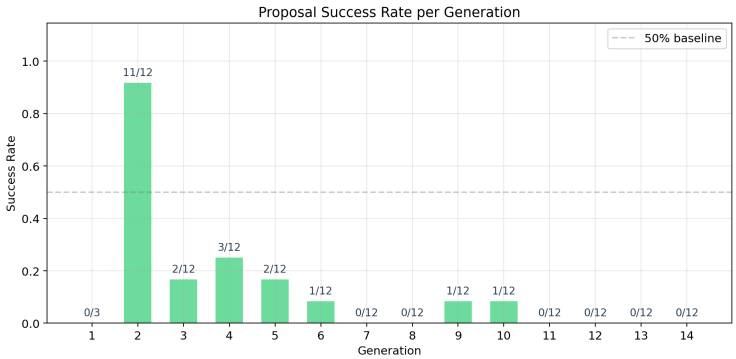


Figure 24: **SST-5: improving proposals are common early and rare later.** Fraction of proposed children whose validation score exceeds their parent's, per step: $11/12$ at step 2, at most $3/12$ from step 3, and zero in most steps after step 6.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim an edit-graph formalization, selective mutability with a price-of-freezing bound, an elitist selector with diversity slots and its guarantees, reachability and convergence results under a stated proposer condition, a length-regularization bias bound, and empirical results: wins on four of seven public benchmarks and matches on three (a margin of at most 0.5 pp), plus a contract stress test. These are substantiated in Sections 2 and 4 and, in full, Sections E to F.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 states the main limitations: the guarantees are conditional on proposal coverage and on a submodularity ratio for which there is no estimator; freezing preserves a contract's text but not its effect (compliance falls from 96.5% to 66.8% in Section 4); a misspecified reward is the dominant failure mode; baselines are taken as published. Section E states what is and is not provable, and Section F reports null and negative ablation results (e.g. momentum).

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Every theorem-like statement is given with its assumptions in Section E (main-body statements are restated there), and complete proofs are in Section E. Results that are not proved are labelled as conjectures.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 specifies all benchmarks, target models and the published protocols reproduced without modification; Algorithm 1 gives the full procedure; Section F give protocols, splits and hyperparameters, and Section A gives optimized prompts.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code is not released with this version. The paper provides full algorithmic pseudocode, all hyperparameters and configuration details sufficient to reimplement the method, and the contract used in the stress test is given verbatim; all benchmarks used are public.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 specifies benchmarks, target models and protocols; Algorithm 1 lists every hyperparameter (beam width W, diversity slots D, quality-gate ratio r, proposals per candidate k, horizon, momentum window, patience); data splits and their roles are defined in Section C and per-experiment settings are in Section F.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The contract stress test and the momentum ablation report mean $\pm$ s.d. over three paired seeds, with paired per-seed differences; the critique ablation reports 95% bootstrap CIs over ten optimizer models. Benchmark comparisons use baselines as published, which report no seed variance, so margins of at most 0.5 pp are reported as matches rather than wins. Single-run results are identified as such.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments use cloud-hosted LLM APIs rather than local GPU compute. Algorithm 1 determines the number of LLM calls per step, and Section F reports measured inference cost.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses publicly available benchmarks (SST-5, GPQA, ARC, BBH, GSM8k, AIME-2025, PUPA), does not involve human subjects, and does not raise dual-use concerns. The method optimizes prompts for reasoning and classification tasks and does not generate harmful content.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts (reducing manual effort and fragility of prompt engineering, enabling faster iteration cycles and more reliable deployments via selective mutability preserving safety contracts) and negative impacts (SPLICE optimizes whatever reward function it is given; if evaluation data contains biases, the optimized prompt may amplify those biases). The selective mutability mechanism itself is presented as a safeguard against unsafe prompt drift.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release pre-trained models, scraped datasets, or other high-risk assets. The contribution is an optimization algorithm that operates on existing LLM APIs. The optimized prompts are task-specific text strings without inherent misuse risk beyond what the underlying LLMs already present.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets are properly cited: SST-5 [SPW+13], GPQA [RHS+23], GSM8k [CKB+21], ARC [CCE+18], BBH [SRR+23], AIME-2025 [BDP+25], PUPA [SRK+25], HelpSteer2 [Wang et al., 2024b]. All prior methods are cited where techniques are adapted. The LLM APIs are accessed through their official providers.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: No new assets (code, datasets, or models) are released with this submission. The algorithmic contribution is fully described via pseudocode (Algorithms 1–3) and formal definitions in the main text and appendix.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. All evaluations are automated using standard benchmark metrics.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects research. All experiments are conducted using automated evaluation of LLM outputs against benchmark datasets.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to the SPLICE methodology and are fully described. The proposal agent generates improved prompt variants (Algorithm 1, step iv). The critique agent produces dual textual gradients (Algorithm 1, step ii). Multiple system models across capability tiers serve as the target models being optimized (Section 3.1). The paper explicitly describes the role of each LLM component.