CREDIT WHERE THE CUT HAPPENS: CANDIDATE CONTRACTION FOR REPOSITORY-LEVEL CODE LOCALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Repository-level code localization advances mainly by exclusion: an agent rules out code that looks relevant but lies off the fix path. Existing training signals cannot credit such a move: with a free-form answer, a discarded location never appears. We introduce C3L (Candidate Contraction for Code Localization), which makes exclusions first-class learning events. C3L binds the prediction to an explicit survivor set that the agent contracts monotonically from read-only evidence, so every exclusion becomes an irreversible, observable state transition. Two potentials over this state price each cut on the turn it is made: a logarithmic weighted volume rewards effective contraction, and a reachable localization ceiling charges coverage loss the moment a necessary location is discarded, with no learned verifier and no extra rollouts. Because the survivor set summarizes the remaining search, C3L compares turns across rollouts by survivor-set signature even when their histories share nothing, and an execution-verified label cache credits valid repairs outside the reference patch. With a 4B backbone, C3L reaches 59.61/47.60/55.00 mean F1 on SWE-bench Verified, SWE-bench Pro (Python), and LocBench, exceeding the strongest retrained same-size baseline by 6.51/7.22/6.21 points (every 95% CI excludes zero). It outscores a frozen 30B-A3B agent and resolves 40.40% of Verified issues with a fixed repair backend, against 37.33% for the strongest baseline. The gain has three measured sources: the exclusion-aware potentials add 5.9 points over terminal-reward training, grouping turns by survivor signature beats grouping them by action history by 1.35 points, and delivering credit per turn adds 2.11 points at identical total return.

1 INTRODUCTION

Resolving a software issue starts with finding where to edit: a locator must return the few files, classes, and functions that a fix touches, and a repairer can only change what it is shown (Jimenez et al., 2024; Xia et al., 2025). With the repair backend fixed, the one-shot resolve rate on SWE-bench Verified moves from 34.20% to 46.20% depending only on the locations it receives (§ 4.3). Recent work therefore trains small models to search repositories with read-only tools and reinforcement learning (Ma et al., 2025; Zhang et al., 2025; Sutawika et al., 2026; Xu et al., 2026; Kim et al., 2026).

Such search is less about finding plausible code than about ruling it out. A report about a failing asynchronous request retrieves the client that must change, but also a similarly named synchronous adapter, a shared base class, and their callers. The agent progresses when it reads the adapter, establishes that it lies off the failing path, and drops it. That exclusion discovers nothing, yet it decides whether the remaining search can succeed; a wrong exclusion is just as silent and removes part of every correct answer long before the episode ends.

Existing training signals cannot price these decisions. Terminal F1 spreads one advantage over the whole trajectory (Sutawika et al., 2026), discovery and commitment rewards credit turns that surface or submit a relevant location (Kim et al., 2026), and novelty rewards credit tool calls that return unseen code (Xu et al., 2026). All of them score what the agent found; none asks *after this exclusion, is a viable repair still reachable?* The obstacle is structural: when the answer is a free-

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

form list submitted at the end, an exclusion is never an event, so there is no turn to credit or blame and no state on which two rollouts can be compared.

C3L (Candidate Contraction for Code Localization) makes exclusions observable. Retrievers build a candidate universe from the issue and the pre-fix repository; at each turn the agent reads code and declares which candidates to exclude or keep; exclusions are irreversible; and the **survivor set** left when the agent stops is the prediction (Figure 1). This single change yields three learning signals. Every exclusion becomes a state transition that a retrieval-weighted log volume rewards on the turn it happens. The best F1 that later cuts can still reach becomes a computable function of the survivor set, so a reachable localization ceiling charges coverage loss the moment a necessary location is discarded, with no learned verifier or extra rollouts. And because the survivor set summarizes the residual search, turns from rollouts that read different files can be compared by a survivor-set signature. A training-time cache adds alternative labels whose patches pass the instance's tests; at inference the agent sees only the issue, the repository, and its survivor set.

We train Qwen3-4B and Qwen3-8B locators and retrain three recent RL localization methods under identical data, tools, candidates, and SFT initialization. 4B C3L reaches 59.61, 47.60, and 55.00 mean F1 on SWE-bench Verified, SWE-bench Pro (Python), and LocBench, beating the strongest same-size baseline by 6.51, 7.22, and 6.21 points (every paired 95% CI above zero); it outscores a frozen Qwen3-Coder-30B-A3B agent (59.61 against 54.79) and, with a fixed repair backend, resolves 40.40% of Verified issues against 37.33% for the strongest baseline. The gain has three measured sources: the exclusion-aware potentials add 5.9 points over terminal-reward training, grouping turns by survivor signature beats grouping them by recent action history by 1.35 points (§ 4.4), and delivering the same total return per turn rather than at the end adds 2.11 points. The margin over terminal-reward training grows with the density of look-alike distractors, from 8.70 to 15.05 points (Appendix A.6).

Our contributions are as follows.

- **Exclusions as learnable events.** We bind the prediction to a monotonically contracting survivor set, so every exclusion becomes an observable, irreversible state transition and the reachable localization ceiling becomes a computable function of state.
- **Closed-form per-turn credit.** Two closed-form potentials price contraction and coverage loss, a protocol-defined survivor-set signature compares turns across rollouts, suffix decay follows agent-caused coverage loss, and execution-verified labels credit alternative repairs, with no learned verifier, extra rollouts, or change at inference.
- **Evidence.** Across three benchmarks and two scales, C3L outperforms retrained RL localizers and a frozen 30B-A3B agent and improves repair; a component ladder, a signature-versus-history comparison, an equal-return timing control, and distractor-stratified analyses tie the gain to credit for exclusions.

2 Related Work

Repository-level code localization. SWE-bench and its successors evaluate issue resolution on real repositories (Jimenez et al., 2024; Chowdhury et al., 2024; Deng et al., 2025), and LocBench isolates localization (Chen et al., 2025). Locators rank code by similarity to the issue (Robertson & Zaragoza, 2009; Suresh et al., 2025; Reddy et al., 2026), narrow it in fixed pipeline stages (Xia et al., 2025; Xie et al., 2025), or navigate with tool-using agents (Yang et al., 2024; Wang et al., 2025b) guided by code graphs (Ouyang et al., 2025; Chen et al., 2025; Jiang et al., 2025) or action scheduling (Yu et al., 2025). Trained searchers include ToolTrain (Ma et al., 2025), RepoNavigator (Zhang et al., 2025), SWE-grep and CodeGrep (Pan et al., 2025a; Chen et al., 2026), CodeScout with three-granularity F1 (Sutawika et al., 2026), FuseSearch with novelty rewards (Xu et al., 2026), and A^2Agent with discovery and commitment rewards (Kim et al., 2026). In all of them, narrowing is a fixed stage or an implicit consequence of the submission; C3L makes it the learned decision and credits every cut. Classical fault localization also eliminates, but by fixed rules over failing executions (Wong et al., 2016; Zeller & Hildebrandt, 2002).

Credit assignment for multi-turn LLM agents. Group-relative optimization gives every turn of a trajectory the same advantage (Shao et al., 2024; Ahmadian et al., 2024; Zheng et al., 2025).

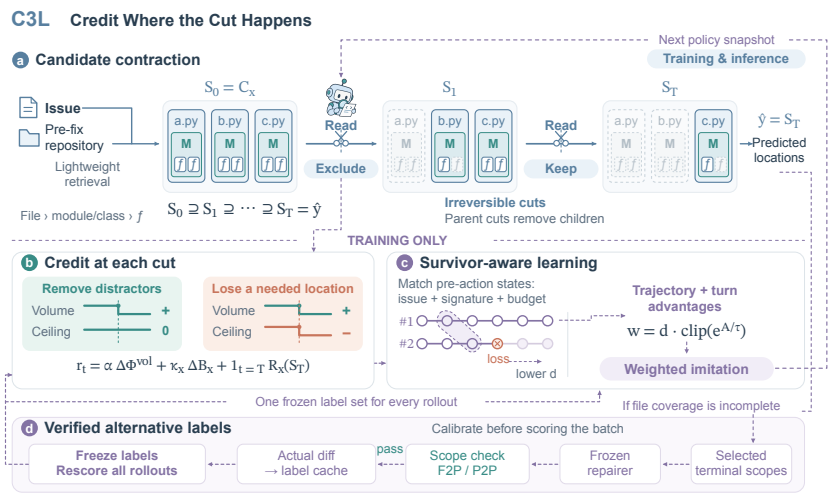


Figure 1: Overview of C3L. **(a)** The agent contracts a retrieved candidate universe by irreversible exclusions and keep restrictions; the final survivors are the prediction (the only panel used at inference). **(b)** Each cut earns a log-volume gain and, if it discards a needed location, a drop in the reachable ceiling. **(c)** Turns are compared within groups keyed by the pre-action survivor signature and budget; fused advantages and the decay factor d set nonnegative token weights. **(d)** Terminal scopes with incomplete file coverage are repaired and tested, and passing diffs become alternative labels before the update.

Finer credit comes from learned verifiers (Lightman et al., 2024; Wang et al., 2024), Monte Carlo resampling (Kazemnejad et al., 2025; Guo et al., 2025), dense or progress rewards (Wei et al., 2025; Wang et al., 2025a), information gain (Wang et al., 2026), tree rollouts (Ji et al., 2026), or step groups keyed by recurring environment states (Feng et al., 2025), history consistency (He et al., 2026), or recent actions (Kim et al., 2026). In C3L the protocol defines the state: process rewards are closed-form potential differences over the survivor set, and turns are grouped by a signature that matches the residual search even when histories share nothing.

Shaping, weighted imitation, and multiple valid repairs. C3L's potentials do not vanish at termination, so unlike policy-invariant shaping (Ng et al., 1999) they deliberately add preferences for contraction and coverage; their telescoping also fixes each trajectory's total return, which lets a control isolate where credit lands. The update follows reward- and advantage-weighted regression and filtered self-training (Peters & Schaal, 2007; Peng et al., 2019; Gulcehre et al., 2023; Dong et al., 2023) with positive weights, because the acceptable-label set grows during training. Building on test-based evaluation (Jimenez et al., 2024) and executable environments (Pan et al., 2025b; Jain et al., 2025; Yang et al., 2025), C3L turns passing repairs into alternative labels.

3 Method

Task. Given an issue x and a pre-fix repository, a locator predicts the locations to change at three granularities $\mathcal{G} = \{f, m, u\}$: files, modules (classes and module-level symbols), and functions. Labels $y = (y_f, y_m, y_u)$ come from the reference patch with the parents needed for hierarchical consistency. A predicted set P_g scores $\mathrm{F1}_g = 2|P_g \cap y_g|/(|P_g| + |y_g|)$ (1 if both sets are empty, 0 if only one is), and $F(P, y)$ averages the three granularities. The agent interacts through read-only tools (search, directory browsing, symbol lookup, code reading); one tool call followed by a candidate decision is the unit of credit. C3L binds the answer to a survivor set (§ 3.1), prices each change of that set (§ 3.2), turns the prices into updates (§ 3.3), and adds labels for alternative repairs (§ 3.4); Algorithm 1 states one iteration and Appendix B.1 gives further details.

3.1 CANDIDATE-CONTRACTION PROTOCOL

Low-cost retrievers (BM25, symbol-name and path matching) form a candidate union from the issue and the pre-fix repository, parsed into entities with containment relations: the candidate universe $C_x = (C_x^f, C_x^m, C_x^u)$ and initial survivor set $S_0 = C_x$, identical in training and inference. At turn t the agent calls a read-only tool and declares an exclusion set X_t and a keep set K_t (identifiers or path patterns); the environment applies, per granularity,

$$S_t = \mathrm{close}\big((S_{t-1} \setminus X_t) \cap K_t\big),$$

with $K_t^g = S_{t-1}^g$ on undeclared granularities, where close removes the children of removed parents. Exclusions reject individual candidates, keeps contract a whole granularity, and $S_t \subseteq S_{t-1}$ always holds. When the agent stops or exhausts its budget, the final survivor set is the prediction $\hat{y} = S_T$; there is no separate answer list. Because removed candidates cannot return, S_t describes exactly what remains reachable and every coverage loss stays attached to the turn that caused it; the agent can still read removed code. A short SFT stage on demonstrations of free submission (*Free*) and contraction (*Prune*) teaches the output format, and all methods start from this checkpoint.

3.2 CANDIDATE CONTRACTION AND COVERAGE LOSS

Weighted candidate volume. Counting candidates would pay for deleting low-relevance code in bulk, so each candidate is weighted by a frozen, label-free retriever score $q_x(a) > 0$ with $\sum_{a \in C_x} q_x(a) = 1$ (Appendix B.3). The volume and the contraction potential are

$$V_x(S) = \sum_{a \in S} q_x(a), \qquad \Phi_x^{\mathrm{vol}}(S) = \log \frac{1 + \varepsilon_v}{V_x(S) + \varepsilon_v},$$

so $\Phi_x^{\mathrm{vol}}(S_0) = 0$ and the contraction reward $r_t^{\mathrm{vol}} = \Phi_x^{\mathrm{vol}}(S_t) - \Phi_x^{\mathrm{vol}}(S_{t-1}) \geq 0$ pays the same for the same proportional cut throughout the search. The weighting pays most for ruling out candidates that look relevant, which is what excluding a look-alike distractor does; its pull toward the reference locations is balanced by the second signal.

Reachable localization ceiling. Let $\mathcal{Y}_x$ be the acceptable labels for issue x, initially the reference label alone, and $\rho_g(S; y') = |S_g \cap y'_g|/|y'_g|$ the coverage of label y'. Later cuts can delete irrelevant candidates but never add new ones, so the best reachable F1 at granularity g is $b_g = 2\rho_g/(1 + \rho_g)$ (1 if $|y'_g| = 0$), and

$$B_x(S) = \max_{y' \in \mathcal{Y}_x} \frac{1}{3} \sum_{g \in \mathcal{G}} b_g(S; y')$$

is the ceiling imposed by the current candidates, an upper bound on reachable F1 that never increases along a trajectory. Coverage loss $r_t^{\mathrm{cap}} = B_x(S_t) - B_x(S_{t-1}) \leq 0$ is zero for an irrelevant exclusion and charges a necessary one on the turn it happens. The turn reward is

$$r_t = \alpha r_t^{\mathrm{vol}} + \kappa_x r_t^{\mathrm{cap}} + \mathbf{1}[t = T] R_x(S_T), \qquad R_x(S_T) = \max_{y' \in \mathcal{Y}_x} F(S_T, y'),$$

where κ_x is higher when the issue, a traceback, or an F2P test points directly at the reference module, and α is set so that no amount of contraction can buy back a complete coverage loss (Appendix B.1).

What the reward prices and where credit lands. Both process rewards are potential differences, so with $\gamma = 1$ they telescope:

$$\sum_{t=1}^{T} r_t = \alpha \Phi_x^{\mathrm{vol}}(S_T) + \kappa_x \big[B_x(S_T) - B_x(S_0)\big] + R_x(S_T).$$

Relative to terminal reward, the potentials add explicit preferences for contraction and coverage. For a fixed total return, paying it per turn or at the end is equivalent for a trajectory-level objective but not for the turn-level comparison of § 3.3: under terminal payment every partial return equals the total, so a turn is ranked partly by coverage lost before it, whereas per-turn payment scores only what follows S_{t-1} and charges each loss once, to the exclusion that caused it. This reward-to-go credit exists for exclusions only because the protocol makes each one a priced state transition; § 4.5 measures its value at identical total return.

3.3 SURVIVOR-AWARE POLICY OPTIMIZATION

Two rollouts can read entirely different files yet leave nearly the same candidates, so C3L compares both whole trajectories and single turns that face similar residual tasks, following within-group comparison on repeated states (Feng et al., 2025) but with the protocol supplying the state. Each iteration freezes $\pi_{\theta_{\text{old}}}$ and samples M rollouts per issue, indexed by $\mathcal{I}_x$. With total return $J_i = \sum_t r_{i,t}$, the trajectory advantage is $A_i^{\text{traj}} = (J_i - \text{mean}_{j\in\mathcal{I}_x} J_j)/(\text{std}_{j\in\mathcal{I}_x} J_j + \varepsilon_a)$.

Survivor-set signature. Exact survivor sets rarely match (28.7% of turns would lack a valid group; Appendix A.5), so the signature $\sigma(S) = (\psi_\ell(S), \nu(S))$ records where survivors sit, $\psi_\ell(S) = \{p \in \mathcal{P}_\ell : \exists a \in S,\ \text{path}(a) \in p\}$ for an ℓ-level directory partition $\mathcal{P}_\ell$, and how much search remains, $\nu(S) = \lceil \log_2(1/V_x(S)) \rceil$. Turn (i,t) joins the group $\mathcal{H}_{i,t}$ of turns with the same issue, pre-action signature $\sigma(S_{i,t-1})$, and remaining-budget bin. The partial return $G_{i,t} = \sum_{k\geq t} r_{i,k}$ telescopes to

$$G_{i,t} = \alpha\big[\Phi_x^{\text{vol}}(S_{i,T_i}) - \Phi_x^{\text{vol}}(S_{i,t-1})\big] + \kappa_x\big[B_x(S_{i,T_i}) - B_x(S_{i,t-1})\big] + R_x(S_{i,T_i}),$$

the further contraction, the ceiling given up, and the final outcome. The turn advantage $\hat{A}_{i,t}$ standardizes $G_{i,t}$ within $\mathcal{H}_{i,t}$ (zero if the group has fewer than two rollouts or no spread), and $A_{i,t} = \omega_0 A_i^{\text{traj}} + \omega_1 \hat{A}_{i,t}$ with $\omega_0 + \omega_1 = 1$.

Suffix decay. Coverage loss belongs to the exclusion that caused it, not to every later token. With the file-level ceiling B_x^f, the pre-action factor $d_{i,t} = B_x^f(S_{i,t-1})/B_x^f(S_{i,0}) \in [0,1]$ (1 when $B_x^f(S_{i,0}) = 0$) shrinks later weights with the coverage already lost; normalizing by the initial ceiling charges retrieval misses to retrieval, and later turns are masked only once no coverage remains.

Update objective. Because $\mathcal{Y}_x$ grows during training, an early low score need not mark a poor trajectory, so C3L uses positive-weight imitation (Peters & Schaal, 2007; Peng et al., 2019) rather than a likelihood-ratio gradient, with no importance ratio or KL term:

$$w_{i,t} = d_{i,t}\ \text{clip}\big(e^{A_{i,t}/\tau}, w_{\min}, w_{\max}\big), \quad \mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{V}_i|}\sum_{n\in\mathcal{V}_i}\text{sg}(w_{i,t(n)})\log\pi_\theta(z_{i,n} \mid h_{i,n}),$$

where $\mathcal{V}_i$ indexes the agent-generated tokens of trajectory i, $t(n)$ is the turn of token $z_{i,n}$, $h_{i,n}$ its prefix, and sg stops gradients.

3.4 TRAINING-TIME ALTERNATIVE-LABEL CACHE

A reference patch is one realized repair; a trajectory that excludes it but keeps another repairable path should not be over-penalized. When a rollout loses complete file coverage ($B_x^f(S_T) < 1$), a frozen repairer generates a patch restricted to S_T, and if the actual edit passes the scope check and the fail-to-pass (F2P) and pass-to-pass (P2P) tests, its canonical diff locations join $\mathcal{Y}_x$. Because the label records where the repair happened rather than the allowed scope, a larger S_T still lowers precision. Validation is capped at k deduplicated scopes per issue and iteration, $|\mathcal{Y}_x|$ is capped, and all trajectories of an issue are rescored under the frozen updated label set before advantages are formed. The cache exists only during training: official scores use the reference labels, and the main repairer (§ 4.3) takes no part in calibration.

4 EXPERIMENTS

4.1 SETUP

Data and models. From the Python instances of SWE-smith (Yang et al., 2025), with test repositories removed, we build 2,400 training and 267 validation issues. We evaluate on all 500 issues of SWE-bench Verified (Chowdhury et al., 2024), the 266-issue public Python subset of SWE-bench Pro (Deng et al., 2025), and all 560 issues of LocBench V1 (Chen et al., 2025), keeping issues with missing candidates or new entities. The locators are Qwen3-4B-Instruct-2507 (Qwen Team, 2025a) and Qwen3-8B (Qwen Team, 2025b). The frozen Qwen3-Coder-30B-A3B-Instruct (Qwen

Table 1: Repository-level localization under a fixed search budget.

	Verified				Pro-Python			LocBench	
Method	**File**	**Module**	**Func**	**Mean**	**File**	**Func**	**Mean**	**Mean**	**Empty (%)**
Retrieval	26.49	22.33	25.54	24.78	22.83	14.46	16.08	22.00	0.00
30B Agentless-Loc	74.10	41.85	30.62	48.86	59.71	20.33	36.90	43.07	**0.60**
Frozen-30B-Free	**81.96**	46.40	36.01	**54.79**	**66.92**	**24.70**	**42.15**	**48.30**	1.00
Frozen-30B-Prune	72.27	**47.51**	**37.99**	52.59	60.46	24.09	39.83	45.92	0.80
4B Base-Free	48.37	18.92	12.49	26.59	35.47	7.38	18.17	22.93	2.60
4B Base-Prune	43.25	17.23	9.81	23.43	29.88	6.22	15.71	19.81	3.80
4B SFT-Prune	$63.81_{\pm 0.66}$	$39.20_{\pm 1.17}$	$28.70_{\pm 1.26}$	$43.91_{\pm 1.03}$	$48.07_{\pm 0.96}$	$16.11_{\pm 1.63}$	$29.64_{\pm 1.38}$	$35.10_{\pm 1.16}$	$0.80_{\pm 0.20}$
4B Terminal-Free	$66.96_{\pm 0.82}$	$40.50_{\pm 1.24}$	$31.10_{\pm 1.38}$	$46.19_{\pm 1.15}$	$51.39_{\pm 1.03}$	$18.50_{\pm 1.79}$	$32.10_{\pm 1.48}$	$37.99_{\pm 1.27}$	$0.53_{\pm 0.31}$
4B Terminal-Prune	$68.41_{\pm 0.71}$	$42.19_{\pm 1.18}$	$32.80_{\pm 1.25}$	$47.80_{\pm 1.04}$	$52.63_{\pm 0.96}$	$19.89_{\pm 1.66}$	$33.20_{\pm 1.35}$	$39.39_{\pm 1.16}$	$0.47_{\pm 0.46}$
4B A^{2}Agent-Matched	$71.33_{\pm 0.54}$	$45.65_{\pm 0.93}$	$37.22_{\pm 1.02}$	$51.40_{\pm 0.83}$	$56.59_{\pm 0.77}$	$27.68_{\pm 1.32}$	$40.38_{\pm 1.11}$	$46.00_{\pm 0.95}$	$0.53_{\pm 0.23}$
4B CodeScout-Matched	$66.97_{\pm 0.67}$	$44.51_{\pm 1.05}$	$35.20_{\pm 1.13}$	$48.90_{\pm 0.95}$	$50.34_{\pm 0.89}$	$27.70_{\pm 1.49}$	$37.87_{\pm 1.24}$	$43.79_{\pm 1.04}$	$0.27_{\pm 0.12}$
4B FuseSearch-Seq	$73.20_{\pm 0.57}$	$45.30_{\pm 0.86}$	$40.80_{\pm 0.95}$	$53.10_{\pm 0.79}$	$57.79_{\pm 0.67}$	$27.21_{\pm 1.24}$	$39.55_{\pm 1.05}$	$48.79_{\pm 0.87}$	$0.27_{\pm 0.12}$
4B C3L-core	$78.98_{\pm 0.46}$	$51.73_{\pm 0.73}$	$44.20_{\pm 0.79}$	$58.30_{\pm 0.66}$	$66.56_{\pm 0.59}$	$32.39_{\pm 1.00}$	$46.16_{\pm 0.82}$	$53.90_{\pm 0.71}$	$0.27_{\pm 0.23}$
4B C3L	$\mathbf{80.03_{\pm 0.38}}$	$\mathbf{52.98_{\pm 0.67}}$	$\mathbf{45.81_{\pm 0.72}}$	$\mathbf{59.61_{\pm 0.59}}$	$\mathbf{68.00_{\pm 0.56}}$	$\mathbf{34.31_{\pm 0.94}}$	$\mathbf{47.60_{\pm 0.78}}$	$\mathbf{55.00_{\pm 0.65}}$	$\mathbf{0.00_{\pm 0.00}}$
8B Terminal-Prune	$69.38_{\pm 0.69}$	$43.61_{\pm 1.03}$	$34.60_{\pm 1.15}$	$49.19_{\pm 0.95}$	$54.22_{\pm 0.83}$	$21.59_{\pm 1.47}$	$34.57_{\pm 1.19}$	$40.89_{\pm 1.04}$	$1.00_{\pm 0.35}$
8B A^{2}Agent-Matched	$72.12_{\pm 0.53}$	$46.79_{\pm 0.83}$	$39.81_{\pm 0.90}$	$52.90_{\pm 0.75}$	$58.62_{\pm 0.68}$	$29.16_{\pm 1.16}$	$41.80_{\pm 0.96}$	$47.61_{\pm 0.82}$	$0.40_{\pm 0.20}$
8B C3L-core	$80.48_{\pm 0.39}$	$53.21_{\pm 0.62}$	$46.00_{\pm 0.70}$	$59.90_{\pm 0.57}$	$68.75_{\pm 0.51}$	$34.80_{\pm 0.91}$	$47.81_{\pm 0.73}$	$55.49_{\pm 0.60}$	$0.27_{\pm 0.12}$
8B C3L	$\mathbf{81.48_{\pm 0.32}}$	$\mathbf{54.59_{\pm 0.56}}$	$\mathbf{47.51_{\pm 0.62}}$	$\mathbf{61.19_{\pm 0.50}}$	$\mathbf{69.89_{\pm 0.48}}$	$\mathbf{36.60_{\pm 0.84}}$	$\mathbf{49.18_{\pm 0.70}}$	$\mathbf{56.71_{\pm 0.56}}$	$\mathbf{0.13_{\pm 0.23}}$

Note. F1 scores are percentages. Trainable configurations report $\mathrm{mean}_{\pm \mathrm{SD}}$ across 3 seeds; untrained and frozen configurations report single values. Within each block, the best F1 and the lowest empty rate are bold and the second best underlined (ties all underlined). Pro: Pro-Python; LocBench: LocBench V1. Empty is the empty-prediction rate on Verified. For space, Pro-Python omits module F1 and LocBench reports Mean F1 only; all granularities, precision, recall, and output sizes appear in Figures A.16–A.18.

Team, 2025c) provides SFT demonstrations, a larger-model reference, and the downstream repairer, and Qwen2.5-Coder-32B-Instruct (Qwen Team, 2024) performs training-time calibration (Appendix B.2).

Training and budget. All methods share candidates (at most 300 files and 2,000 functions with their parents), four read-only tools, a budget of 20 tool calls and 8,192 output tokens, and one Free/Prune SFT checkpoint per backbone and seed, followed by 3 sampling-and-training iterations with 8 rollouts per issue and 2 epochs each. Checkpoints are selected on validation, and evaluation uses one greedy trajectory scored against reference labels (Appendices B.3–B.5). The initial candidates fully cover the reference on 74.8%, 58.3%, and 64.8% of issues (Table A.1); the reference ceiling B_x^{ref} is B_x computed with the fixed reference label.

Baselines and scoring. Retrieval, Agentless-Loc, and untrained Base and Frozen-30B runs are references. Terminal-Free and Terminal-Prune train on terminal F1 under C3L's objective; C3L-core adds every process component, and full C3L adds the cache. A^{2}Agent-Matched (Kim et al., 2026), CodeScout-Matched (Sutawika et al., 2026), and FuseSearch-Seq (Xu et al., 2026) are retrained in the shared setting with their own training signals and, for the latter two, their GSPO or GRPO objectives on an on-policy schedule with the same number of optimizer steps as C3L (Appendices B.4 and B.6). Trainable configurations report the mean and sample SD over 3 seeds; paired 95% CIs come from 10,000 issue-level bootstrap resamples (Appendix B.7). Repair uses a fixed Agentless (Xia et al., 2025) backend with one greedy patch per issue.

4.2 Localization Performance

C3L outperforms every retrained baseline at every reported granularity on all three benchmarks (Table 1).

4B C3L beats the strongest same-size baseline on each benchmark by 6.51, 7.22, and 6.21 points (95% CIs [4.94, 8.11], [5.18, 9.29], [4.75, 7.66]); each gap exceeds six times the larger seed SD, and a two-level bootstrap over seeds and issues keeps all three intervals above zero (Appendix B.7). At 8B, C3L leads A^{2}Agent-Matched on Verified by 8.29 points. The margin comes neither from the objective nor from re-implementation: A^{2}Agent-Matched shares C3L's objective, so its 8.21-point gap at 4B isolates the training signal; C3L exceeds A^{2}Agent's published results (52.72 on Verified, 41.93 on Pro-Python) by 6.89 and 5.67 points (Appendix A.4); and trained with GRPO under the baselines' schedule, C3L-core still reaches 55.25% on Verified, above FuseSearch-Seq and CodeScout-Matched (Figure A.6).

Table 2: One-shot repair on SWE-bench Verified with a fixed Qwen3-Coder-30B backend.

	Localization		Repair		By edit size	
Method	**Mean F1**	**Coverage**	**Resolve**	**Truncation**	**Single**	**Multi**
4B SFT-Prune	$43.91_{\pm 1.03}$	$70.13_{\pm 1.55}$	$35.67_{\pm 0.50}$	$31.47_{\pm 0.70}$	$37.07_{\pm 0.73}$	$30.53_{\pm 1.08}$
4B Terminal-Free	$46.19_{\pm 1.15}$	$74.00_{\pm 2.31}$	$35.53_{\pm 0.50}$	$31.87_{\pm 2.21}$	$36.73_{\pm 0.53}$	$31.15_{\pm 0.54}$
4B Terminal-Prune	$47.80_{\pm 1.04}$	$74.47_{\pm 1.21}$	$36.53_{\pm 0.70}$	$28.47_{\pm 0.90}$	$38.25_{\pm 0.96}$	$30.22_{\pm 1.08}$
4B A^{2}Agent-Matched	$51.40_{\pm 0.83}$	$77.40_{\pm 0.92}$	$37.33_{\pm 0.64}$	$31.93_{\pm 1.55}$	$39.02_{\pm 0.78}$	$31.15_{\pm 0.54}$
4B CodeScout-Matched	$48.90_{\pm 0.95}$	$68.20_{\pm 0.92}$	$34.20_{\pm 0.20}$	$\underline{23.40}_{\pm 0.72}$	$35.28_{\pm 0.39}$	$30.22_{\pm 1.43}$
4B FuseSearch-Seq	$53.10_{\pm 0.79}$	$71.47_{\pm 0.50}$	$36.73_{\pm 0.31}$	$\mathbf{20.07}_{\pm \mathbf{0.90}}$	$37.57_{\pm 0.39}$	$33.64_{\pm 0.00}$
4B C3L-core	$\underline{58.30}_{\pm 0.66}$	$\underline{81.00}_{\pm 0.72}$	$\underline{39.87}_{\pm 0.70}$	$23.80_{\pm 1.59}$	$\underline{41.48}_{\pm 0.88}$	$\underline{33.96}_{\pm 0.54}$
4B C3L	$\mathbf{59.61}_{\pm \mathbf{0.59}}$	$\mathbf{82.67}_{\pm \mathbf{0.50}}$	$\mathbf{40.40}_{\pm \mathbf{0.53}}$	$24.80_{\pm 0.40}$	$\mathbf{41.98}_{\pm \mathbf{0.67}}$	$\mathbf{34.58}_{\pm \mathbf{1.62}}$
8B Terminal-Prune	$49.19_{\pm 0.95}$	$73.73_{\pm 0.23}$	$36.47_{\pm 0.64}$	$\mathbf{26.67}_{\pm \mathbf{0.58}}$	$38.08_{\pm 1.03}$	$\underline{30.53}_{\pm 1.08}$
8B A^{2}Agent-Matched	$\underline{52.90}_{\pm 0.75}$	$\underline{79.27}_{\pm 0.90}$	$\underline{37.07}_{\pm 0.12}$	$33.20_{\pm 0.40}$	$\underline{39.02}_{\pm 0.15}$	$29.91_{\pm 0.00}$
8B C3L	$\mathbf{61.19}_{\pm \mathbf{0.50}}$	$\mathbf{87.13}_{\pm \mathbf{1.72}}$	$\mathbf{41.80}_{\pm \mathbf{0.60}}$	$\underline{27.60}_{\pm 1.78}$	$\mathbf{43.34}_{\pm \mathbf{0.53}}$	$\mathbf{36.14}_{\pm \mathbf{1.95}}$
Reference-locations	100.00	100.00	46.20	19.80	46.82	43.93

Note. Scores are percentages; $\mathrm{mean}_{\pm \mathrm{SD}}$ across 3 locator seeds, and the reference-input row is deterministic. Within each block, the best mean F1, coverage, and resolve rate are bold and the second best underlined; for truncation the lowest is bold. Coverage is complete file coverage; Truncation is the fraction of issues whose localization context exceeds the repairer's input budget. Single and Multi are resolve rates on the 393 single-file and 107 multi-file issues.

C3L is also more compact and beats larger models. On Verified it raises function precision over A^{2}Agent-Matched from 33.59% to 47.51% and recall from 47.56% to 50.51% while predicting fewer files (1.46 against 1.62), and it never returns an empty prediction; untrained Base-Prune scores below Base-Free, so the protocol alone does not produce this. 4B C3L surpasses the frozen 30B-A3B agent (59.61% against 54.79%), mainly through function F1 (+9.80), and the Agentless pipeline on the same 30B model by 10.75 points. On the 374 Verified issues fully covered at the start it reaches 63.01% (Appendix A.1); Appendix A.9 analyzes its losses against FuseSearch-Seq.

4.3 Downstream Repair

Better localization translates into more resolved issues. Every locator feeds the same frozen Qwen3-Coder-30B-A3B repair backend with identical packing, budget, and tests (Table 2).

4B C3L resolves 40.40% of the 500 Verified issues, 3.07 points above A^{2}Agent-Matched (95% CI [1.53, 4.67]) and 3.87 above Terminal-Prune ([2.33, 5.47]); 8B C3L reaches 41.80%, 4.73 above A^{2}Agent-Matched ([3.20, 6.27]). Reference locations reach 46.20%, so localization still has room to convert into repairs. Complete coverage and compact context connect the two: relative to A^{2}Agent-Matched, 4B C3L raises complete file coverage from 77.40% to 82.67% and lowers truncation from 31.93% to 24.80%, whereas FuseSearch-Seq truncates least (20.07%) but covers only 71.47% and resolves 36.73%. The gain is largest on the 107 multi-file issues, where several locations must survive together: 6.23 points at 8B ([3.12, 9.35]) and 3.43 at 4B ([0.31, 6.54]). C3L-core already reaches 39.87% and the cache adds 0.53; Appendices A.2 and A.3 analyze calibration and a second repairer.

4.4 Component Contributions

Every component contributes. We build C3L one component at a time under the same objective (Table 3); the first six stages use fixed reference labels. To track wrong exclusions we report the prune error rate $\mathrm{PER} = |\mathcal{D}_0^f|^{-1} \sum_{x \in \mathcal{D}_0^f} \mathbf{1}[y_f \not\subseteq S_T^f]$, the fraction of the 467 Verified issues whose reference files are fully covered at the start ($\mathcal{D}_0^f$) that end without a reference file (for Free, the submitted files replace S_T^f).

The protocol alone adds 1.61 points (95% CI [0.23, 3.02]); the four process components add a further 10.50 on Verified and 12.97 on Pro-Python ([10.79, 15.13]), and every paired interval is positive (Table B.6; Appendix A.4 repeats this under GRPO). The potentials are complementary: volume reward alone shrinks predictions from 1.58 to 1.28 files but raises PER from 20.27% to

Table 3: Incremental construction of C3L under the same imitation-weighted objective.

	Mean F1 (%)		Verified state			
Setting	**Verified**	**Pro**	**PER (%)**	**Ref. ceiling**	**Files**	**Empty (%)**
Terminal-Free	$46.19_{\pm 1.15}$	$32.10_{\pm 1.48}$	$20.77_{\pm 2.47}$	$0.6292_{\pm 0.0234}$	$1.67_{\pm 0.05}$	$0.53_{\pm 0.31}$
+ Contraction protocol (Terminal-Prune)	$47.80_{\pm 1.04}$	$33.20_{\pm 1.35}$	$20.27_{\pm 1.29}$	$0.6298_{\pm 0.0144}$	$1.58_{\pm 0.04}$	$0.47_{\pm 0.46}$
+ Volume reward	$49.40_{\pm 0.94}$	$35.21_{\pm 1.26}$	$28.41_{\pm 1.01}$	$0.5754_{\pm 0.0118}$	$1.28_{\pm 0.02}$	$1.60_{\pm 0.87}$
+ Coverage loss	$53.70_{\pm 0.84}$	$40.52_{\pm 1.02}$	$18.56_{\pm 0.87}$	$0.6541_{\pm 0.0137}$	$1.41_{\pm 0.02}$	$0.67_{\pm 0.12}$
+ Turn advantage (survivor signature)	$56.49_{\pm 0.74}$	$44.00_{\pm 0.92}$	$15.27_{\pm 1.80}$	$0.6907_{\pm 0.0110}$	$1.42_{\pm 0.05}$	$0.53_{\pm 0.50}$
+ Suffix decay (C3L-core)	$58.30_{\pm 0.66}$	$46.16_{\pm 0.82}$	$13.28_{\pm 0.77}$	$0.7096_{\pm 0.0072}$	$1.42_{\pm 0.02}$	$0.27_{\pm 0.23}$
+ Label cache (C3L)	$59.61_{\pm 0.59}$	$47.60_{\pm 0.78}$	$11.49_{\pm 0.54}$	$0.7268_{\pm 0.0090}$	$1.46_{\pm 0.00}$	$0.00_{\pm 0.00}$

Note. $\text{mean}_{\pm \text{SD}}$ across 3 seeds; Pro is Pro-Python. State diagnostics are for Verified: Ref. ceiling is the terminal reference ceiling (initially 0.9380 for every row), Files the mean number of predicted files, and Empty the empty-prediction rate. For Terminal-Free, PER measures omissions in the submitted prediction.

28.41%; adding coverage loss lowers PER to 18.56% and restores the ceiling while keeping 1.41 files, for +4.29 points ([2.94, 5.71]).

Credit placement then improves the quality of each cut. The turn-level advantage adds 2.80 points at an unchanged file count, and the survivor signature is what makes it work: in two-seed scans on the C3L-core base, issue-level groups add only 0.25 points over trajectory-only advantages and history groups (the A^{2}Agent key) add 1.39, whereas signature groups add 2.79; a three-seed rerun confirms that the signature beats the history key by 1.35 points on Verified (95% CI [0.19, 2.51]) and 1.19 on Pro-Python, lowering PER from 15.38% to 13.28% (Appendix A.5). Suffix decay adds 1.81 points, and the cache adds 1.31 ([-0.09, 2.70]) and lowers PER to 11.49%; in an execution audit, 20.03% of triggered trajectories hide a test-passing alternative repair (Appendix A.2).

4.5 Search Dynamics, Reward Timing, and Budget

C3L contracts fast while keeping coverage (Figure 2a–d). After five calls on Verified it retains 8.17% of the initial weighted volume at a reference ceiling of 0.9029, against 16.78% and 0.7134 for Terminal-Prune; setting $\kappa = 0$ cuts the volume to 6.77% but collapses the ceiling to 0.6556, with 90.5% of that run's ceiling loss in the first five calls. Pro-Python shows the same pattern, so coverage feedback matters most where contraction is steepest.

Where credit lands is worth 2.11 points on its own (Figure 2e–f). We fix every reward term, disable the cache, and vary only timing and the advantage level; terminal delivery pays $r'_T = J$ and nothing before, so both deliveries have identical total return. Under trajectory-only advantages they coincide exactly and share one checkpoint (Appendix A.5). With the fused advantage, per-turn delivery gains 2.11 points on Verified ([0.94, 3.28]) and 2.29 on Pro-Python ([0.83, 3.75]) and lowers PER from 15.85% to 13.28% and from 42.72% to 35.40%: it charges each coverage loss to the exclusion that caused it, and the policy learns to keep coverage at the decision itself (cases in Appendix A.8).

C3L's advantage is largest under tight budgets (Figure 2g–j). With 5 calls it reaches 50.47% on Verified and 38.70% on Pro-Python, above Terminal-Prune with 20 calls; cutting the budget from 20 to 5 costs C3L 15.3% of its Mean F1 and Terminal-Prune 37.2%. With 20 allowed calls, C3L stops after 13.35 calls against 17.29, cutting input from 412.7K to 285.8K tokens and latency from 305.0 to 265.2 seconds despite longer declarations (6.23K against 4.14K output tokens); Appendix A.7 covers equal-compute training, data scaling, and parallel tools.

5 Conclusion

Repository-level localization advances by ruling code out, yet existing training signals reward only what an agent finds. C3L binds the prediction to a monotonically contracting survivor set, so every exclusion becomes an observable state transition that a weighted log volume and a reachable ceiling price on the turn it happens, and survivor-set signatures compare decisions that face the same resid-

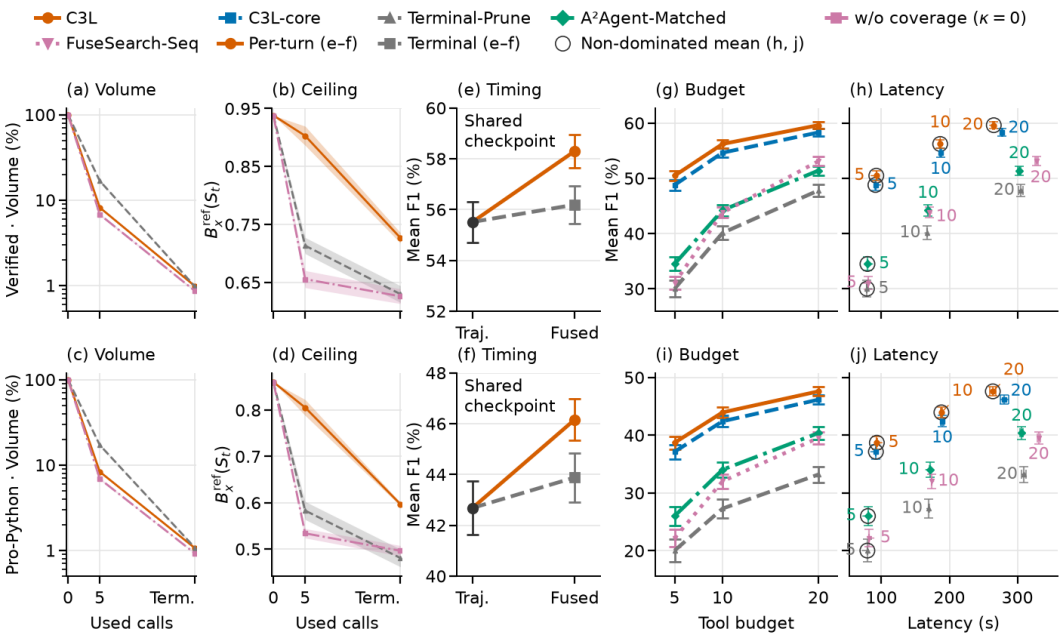


Figure 2: Search dynamics, reward timing, and budget efficiency (top: Verified; bottom: Pro-Python). **(a–d)** Remaining weighted volume (% of initial, log axis) and reference ceiling B_x^{ref} at the start, after 5 tool calls, and at termination (Term.); $\kappa = 0$ is full C3L without the coverage term. **(e–f)** Per-turn versus terminal delivery at equal total return, cache disabled, with trajectory-only (Traj.) or fused advantages; the two trajectory-only runs share one checkpoint. **(g–j)** Quality versus allowed tool budget (5/10/20) and latency; rings mark non-dominated means. Bands and whiskers: $\text{mean}_{\pm\text{SD}}$ over 3 seeds.

ual search. Under matched data, tools, and initialization, a 4B C3L policy outperforms retrained RL localizers on three benchmarks, surpasses a frozen 30B-A3B agent, and improves one-shot repair, with gains from what the reward prices, how turns are compared, and where credit lands. Whenever an agent searches by elimination, making the state of that elimination explicit lets each irreversible decision receive its own credit.

6 Limitations

Irreversibility is what lets the ceiling price a cut; allowing restoration lowers Verified Mean F1 from 59.61 to 54.86 (Appendix A.5), so settings that need undo require a different coverage accounting. Contraction cannot add locations missing from the initial candidates, so new entities remain bounded by retrieval, and the candidate cap still binds on LocBench, where a larger cap adds 4.98 points (Appendix A.1). The label cache relies on positive-weight updates: its 1.31-point increment is not significant and turns slightly negative under GRPO, and 7 of 100 audited labels pass only because of weak tests (Appendices A.2 and A.4). The frozen 30B-A3B reference has about 3B active parameters and was not trained for this protocol, and the 8B locator comes from an earlier Qwen3 release than the 4B model, although both scales rank all methods identically. All data are Python, and the gain needs at least one full sampling-and-training iteration: at one third of the default compute, C3L trails terminal-reward training by 1.85 points (Appendix A.7).

AI USE STATEMENT

In this work, we used large language models (LLMs) to polish the writing of the manuscript, that is, to edit text for grammar, wording, and readability. We have not used generative AI tools for any task whose disclosure is required by the ICLR 2027 AI Policy, including formulating hypotheses, designing the method or experiments, implementing methods, generating or cleaning data, and analyzing or interpreting results; theoretical claims, proofs, and translation are not applicable to this work. We have reviewed all AI-assisted edits and verified that each revised passage preserves the original technical content, claims, and reported numbers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All models, training data, and benchmarks used in this work are publicly available: the Qwen3 and Qwen2.5-Coder checkpoints, the Python subset of SWE-smith, SWE-bench Verified, the public Python subset of SWE-bench Pro, and LocBench V1 (§ 4.1). Appendix B.2 documents the repository-level split, deduplication, and label extraction, including the treatment of newly added entities and empty labels. Appendix B.3 specifies candidate construction, the frozen candidate weights, the catalog format, the action format, hierarchical closure, and the stopping rules, so the shared interaction protocol can be reimplemented as described. Appendix B.4 lists the shared SFT procedure, the training and inference configuration, and every method parameter. Appendix B.5 gives the anchor strata for κ_x, the cache trigger, the validation budget, the scope check, and the fixed downstream repair procedure. Appendix B.6 states how each baseline is adapted and retrained under the shared setting, and Algorithm 1 summarizes one full sampling-and-training iteration. Appendix B.7 defines every metric, the seed-level replication, the paired bootstrap behind all confidence intervals, and the accounting of training and inference cost.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Wuya Chen, Yihao Yang, Yang Cao, and Yue Lin. CodeGrep: An RL-trained retrieval agent for LLM coding agents. *arXiv preprint arXiv:2608.05886*, 2026. URL https://arxiv.org/abs/2608.05886.

Zhaoling Chen, Robert Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. LocAgent: Graph-guided LLM agents for code localization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8697–8727, 2025. doi: 10.18653/v1/2025.acl-long.426. URL https://aclanthology.org/2025.acl-long.426/.

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified. OpenAI, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025. URL https://arxiv.org/abs/2509.16941.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2304.06767.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2505.10978.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023. URL https://arxiv.org/abs/2308.08998.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2505.23564.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.22817.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=7evvwwdo3z.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.21240.

Zhonghao Jiang, Xiaoxue Ren, Meng Yan, Wei Jiang, Yong Li, and Zhongxin Liu. Issue localization via LLM-driven iterative code graph searching. In *Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering*, 2025. URL https://arxiv.org/abs/2503.22424.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2410.01679.

Doyeon Kim, Suyoung Bae, Yumin Lee, and Jee-Hyong Lee. A^{2}Agent: Action-aware reinforcement learning for repository-level code localization agents. *arXiv preprint arXiv:2608.29831*, 2026. URL https://arxiv.org/abs/2608.29831. Accepted to EMNLP 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.20050.

Zexiong Ma, Chao Peng, Qunhong Zeng, Pengfei Gao, Yanzhen Zou, and Bing Xie. Tool-integrated reinforcement learning for repo deep search. *arXiv preprint arXiv:2508.03012*, 2025. URL https://arxiv.org/abs/2508.03012.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, 1999.

Siru Ouyang, Wenhao Yu, Kaixin Ma, Zilin Xiao, Zhihan Zhang, Mengzhao Jia, Jiawei Han, Hongming Zhang, and Dong Yu. RepoGraph: Enhancing AI software engineering with repository-level code graph. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.14684.

Ben Pan, Carlo Baronio, Albert Tam, Pietro Marsella, Mokshit Jain, Daniel Chiu, Swyx, and Silas Alberti. Introducing SWE-grep and SWE-grep-mini: RL for multi-turn, fast context retrieval. Cognition blog, 2025a. URL https://cognition.com/blog/swe-grep.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025b. URL https://arxiv.org/abs/2412.21139.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. URL https://arxiv.org/abs/1910.00177.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 745–750, 2007. doi: 10.1145/1273496.1273590.

Qwen Team. Qwen2.5-Coder-32B-Instruct. https://huggingface.co/Qwen/Qwen2.5-Coder-32B-Instruct, 2024.

Qwen Team. Qwen3-4B-Instruct-2507. https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507, 2025a.

Qwen Team. Qwen3-8B. https://huggingface.co/Qwen/Qwen3-8B, 2025b.

Qwen Team. Qwen3-Coder-30B-A3B-Instruct. https://huggingface.co/Qwen/Qwen3-Coder-30B-A3B-Instruct, 2025c.

Revanth Gangi Reddy, Tarun Suresh, JaeHyeok Doo, Ye Liu, Xuan Phi Nguyen, Yingbo Zhou, Semih Yavuz, Caiming Xiong, Heng Ji, and Shafiq Joty. SweRank: Software issue localization with code ranking. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2505.07849.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Tarun Suresh, Revanth Gangi Reddy, Yifei Xu, Zach Nussbaum, Andriy Mulyar, Brandon Duderstadt, and Heng Ji. CoRNStack: High-quality contrastive data for better code retrieval and reranking. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.01007.

Lintang Sutawika, Aditya Bharat Soni, Bharath Sriraam R R, Apurva Gandhi, Taha Yassine, Sanidhya Vijayvargiya, Yuchen Li, Xuhui Zhou, Yilin Zhang, Leander Melroy Maben, and Graham Neubig. CodeScout: An effective recipe for reinforcement learning of code search agents. *arXiv preprint arXiv:2603.17829*, 2026. URL https://arxiv.org/abs/2603.17829.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.14967.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: Reinforcing LLM agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025a. URL https://arxiv.org/abs/2505.20732.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://arxiv.org/abs/2407.16741.

Quan Wei, Siliang Zeng, Chenliang Li, Zhongruo Wang, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via fine-grained reward structure and credit assignment. *arXiv preprint arXiv:2505.11821*, 2025. URL https://arxiv.org/abs/2505.11821.

W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016. doi: 10.1109/TSE.2016.2521368.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025. doi: 10.1145/3715754.

Chengxing Xie, Bowen Li, Chang Gao, He Du, Wai Lam, Difan Zou, and Kai Chen. SWE-Fixer: Training open-source LLMs for effective and efficient GitHub issue resolution. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 1123–1139, 2025. doi: 10.18653/v1/2025.findings-acl.62. URL https://aclanthology.org/2025.findings-acl.62/.

Ke Xu, Siyang Xiao, Ming Liang, Yichen Yu, Zhixiang Wang, Jingxuan Xu, Dajun Chen, Wei Jiang, and Yong Li. FuseSearch: Learning adaptive parallel execution for efficient code localization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2954–2967, 2026. doi: 10.18653/v1/2026.findings-acl.143. URL https://aclanthology.org/2026.findings-acl.143/.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.15793.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 38, 2025. URL https://arxiv.org/abs/2504.21798.

Zhongming Yu, Hejia Zhang, Yujie Zhao, Hanxian Huang, Matrix Yao, Ke Ding, and Jishen Zhao. OrcaLoca: An LLM agent framework for software issue localization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73416–73436, 2025. URL https://proceedings.mlr.press/v267/yu25x.html.

Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002. doi: 10.1109/32.988498.

Zhaoxi Zhang, Yitong Duan, Yanzhi Zhang, Yiming Xu, Zhixiang Wang, Kun Liang, Weikang Li, Jiahui Liang, Deguo Xia, Jizhou Huang, Jiyan He, and Yunfang Wu. One tool is enough: Reinforcement learning for repository-level LLM agents. *arXiv preprint arXiv:2512.20957*, 2025. URL https://arxiv.org/abs/2512.20957.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025. URL https://arxiv.org/abs/2507.18071.

Table A.1: Initial candidate conditions shared by all methods.

Benchmark	Issues	ICR-all (%)	File recall (%)	Reference ceiling (0–1)	Empty module labels (%)
Verified	500	74.8	96.2	0.9380	2.4
Pro-Python	266	58.3	92.1	0.8605	10.9
LocBench V1	560	64.8	90.3	0.8341	8.9

Note. ICR-all is the fraction of issues whose reference entities are fully covered by the initial candidates at all three granularities. File recall and reference ceilings $B_x^{\mathrm{ref}}(S_0)$ are issue-level macro averages; coverage metrics in the paper, including the terminal reference-ceiling loss $B_x^{\mathrm{ref}}(S_0) - B_x^{\mathrm{ref}}(S_T)$, are measured against this shared start.

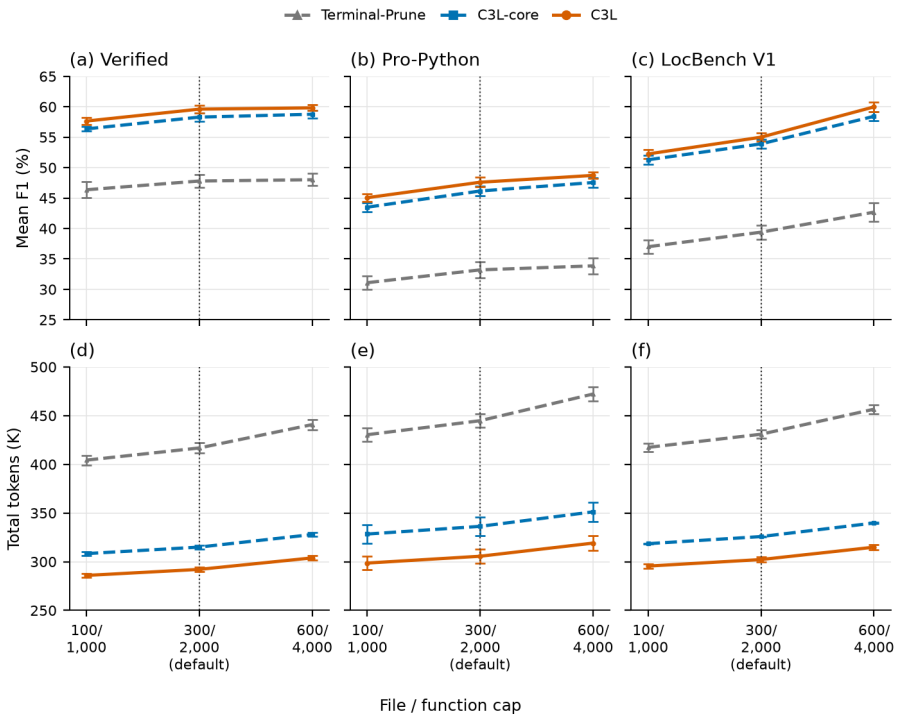


Figure A.1: Sensitivity to candidate-set size. Checkpoints and calibration are frozen at the default 300-file/2,000-function setting. Localization quality shows $\mathrm{mean}_{\pm\mathrm{SD}}$ across 3 seeds; coverage is a fixed construction statistic. Cap settings are ordered categories; the default is marked by a dotted line. Evaluation-subset counts and coverage gaps appear in Table A.3 and Figure A.2.

A Additional Experimental Results

A.1 Candidate Coverage and Scale

Candidate coverage decides which targets a contraction policy can still keep. We freeze the checkpoint and the calibration configuration trained at the default cap of 300 files and 2,000 functions, and we change only the inference cap to 100/1,000, 300/2,000, or 600/4,000. We also separate the subset of entities that are representable before the repair from the subset completely covered by the initial candidates. The first excludes issues that must add a file, a module, or a function. The second further requires the reference labels at all three granularities to lie inside the candidates.

In Figure A.1, the gain from a larger candidate set tracks the retrieval gap. On Verified, ICR-all rises from the default 74.8% to 76.8%, and C3L Mean F1 rises by only 0.21 points. On LocBench V1, coverage rises from 64.8% to 72.1%, and Mean F1 rises by 4.98 points. Expansion helps LocBench because the default cap still misses many representable targets. Table A.2 shows pre-retrieval latency

Table A.2: Candidate construction diagnostics.

	File / function cap		
Initial reference ceiling	**100 / 1,000**	**300 / 2,000 (default)**	**600 / 4,000**
Verified	0.8983	0.9380	0.9424
Pro-Python	0.8261	0.8605	0.8999
LocBench V1	0.7999	0.8341	0.9059
Pre-retrieval latency (s)	6.8	14.6	27.4

Note. Initial reference ceiling on each benchmark and per-issue pre-retrieval latency at three candidate caps. ICR-all and localization quality appear in Figures A.1–A.2.

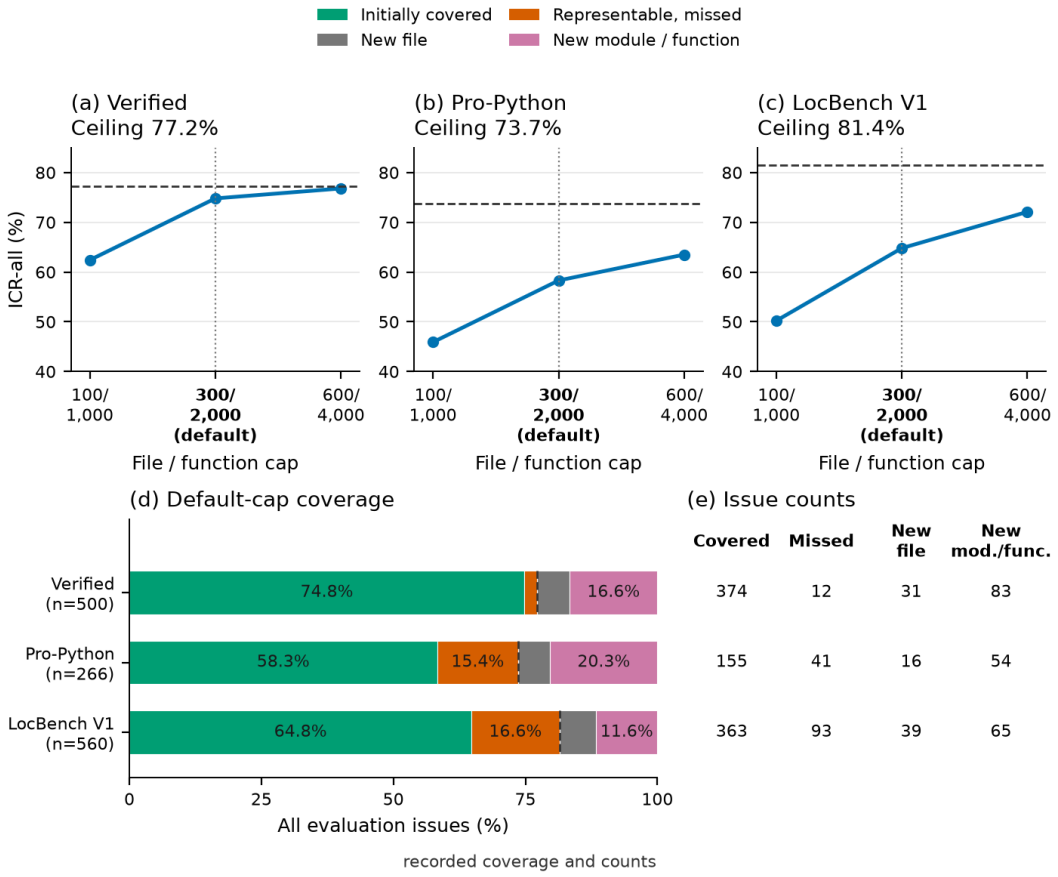


Figure A.2: Initial coverage and representability limits. ICR-all is shown for three file/function caps. Dashed ceilings are 386/500 for Verified, 196/266 for Pro-Python and 456/560 for LocBench V1. Default-cap counts partition each full benchmark into initially complete, representable but missed, new-file and new-module/function issues; the last category excludes new-file issues. Empty labels and empty candidates are separate attributes.

rising from 14.6 seconds at the default cap to 27.4 seconds. A larger candidate set is justified by the coverage it can recover and by what that coverage costs.

Figure A.2 locates the difference in the issue mix. Of the 126 Verified issues that are not initially fully covered, 114 involve a new entity and only 12 are representable but missed by retrieval. LocBench V1 has 93 issues of the latter kind. The representability ceilings are 386/500, 196/266, and 456/560. Enlarging the pre-fix candidate set does not close the gap created by new entities. On the fully covered subsets, Table A.3 reports C3L Mean F1 of 63.01%, 52.28%, and 64.16%, each

Table A.3: Localization on full, representable and initially fully covered evaluation sets.

Benchmark	Scope	Method	Issues	F1 (%) Mean	File	Func	Ceil.
Verified	Full	Terminal-Prune	500	$47.80_{\pm 1.04}$	$68.41_{\pm 0.71}$	$32.80_{\pm 1.25}$	0.9380
	Full	C3L-core	500	$\underline{58.30}_{\pm 0.66}$	$\underline{78.98}_{\pm 0.46}$	$\underline{44.20}_{\pm 0.79}$	0.9380
	Full	C3L	500	$\mathbf{59.61}_{\pm \mathbf{0.59}}$	$\mathbf{80.03}_{\pm \mathbf{0.38}}$	$\mathbf{45.81}_{\pm \mathbf{0.72}}$	0.9380
	Representable	Terminal-Prune	386	$49.75_{\pm 0.95}$	$70.16_{\pm 0.53}$	$34.86_{\pm 1.38}$	0.9941
	Representable	C3L-core	386	$\underline{61.98}_{\pm 0.43}$	$\mathbf{82.71}_{\pm \mathbf{0.33}}$	$\underline{47.95}_{\pm 0.66}$	0.9941
	Representable	C3L	386	$\mathbf{62.58}_{\pm \mathbf{0.32}}$	$\underline{82.64}_{\pm 0.33}$	$\mathbf{49.17}_{\pm \mathbf{0.79}}$	0.9941
	Initially covered	Terminal-Prune	374	$49.99_{\pm 0.99}$	$70.36_{\pm 0.56}$	$35.13_{\pm 1.46}$	1.0000
	Initially covered	C3L-core	374	$\underline{62.50}_{\pm 0.40}$	$\mathbf{83.17}_{\pm \mathbf{0.39}}$	$\underline{48.43}_{\pm 0.70}$	1.0000
	Initially covered	C3L	374	$\mathbf{63.01}_{\pm \mathbf{0.25}}$	$\underline{82.96}_{\pm 0.48}$	$\mathbf{49.62}_{\pm \mathbf{0.76}}$	1.0000
Pro-Python	Full	C3L	266	$47.60_{\pm 0.78}$	$68.00_{\pm 0.56}$	$34.31_{\pm 0.94}$	0.8605
	Representable	C3L	196	$49.85_{\pm 0.53}$	$69.10_{\pm 0.59}$	$36.60_{\pm 1.08}$	0.9297
	Initially covered	C3L	155	$52.28_{\pm 0.69}$	$71.06_{\pm 0.57}$	$38.95_{\pm 1.06}$	1.0000
LocBench V1	Full	C3L	560	$55.00_{\pm 0.65}$	$76.50_{\pm 0.45}$	$42.51_{\pm 0.79}$	0.8341
	Representable	C3L	456	$59.14_{\pm 0.94}$	$79.50_{\pm 0.83}$	$47.26_{\pm 1.07}$	0.9047
	Initially covered	C3L	363	$64.16_{\pm 0.98}$	$82.87_{\pm 0.63}$	$52.93_{\pm 1.12}$	1.0000

Note. $\mathrm{mean}_{\pm \mathrm{SD}}$ across 3 seeds. Counts are specific to each evaluation scope. Representable means representable before repair; Initially covered means initially fully covered. Ceil. is the initial reference ceiling. Within each Verified scope, the best F1 is bold and the second best is underlined. Pro-Python and LocBench report C3L only. Initially fully covered sets have ceiling 1.0000.

Table A.4: Sensitivity to empty module labels.

Benchmark	Empty labels	Empty cand.	Method	Module F1 (%) All	Nonempty
Verified	12	5	Terminal-Free	$40.50_{\pm 1.24}$	$41.16_{\pm 1.43}$
	—	—	Terminal-Prune	$\underline{42.19}_{\pm 1.18}$	$\underline{42.68}_{\pm 1.72}$
	—	—	C3L	$\mathbf{52.98}_{\pm \mathbf{0.67}}$	$\mathbf{53.60}_{\pm \mathbf{0.99}}$
Pro-Python	29	14	Terminal-Free	$26.42_{\pm 1.62}$	$27.26_{\pm 2.25}$
	—	—	Terminal-Prune	$\underline{27.06}_{\pm 1.44}$	$\underline{28.55}_{\pm 1.19}$
	—	—	C3L	$\mathbf{40.48}_{\pm \mathbf{0.84}}$	$\mathbf{40.79}_{\pm \mathbf{0.93}}$
LocBench V1	50	36	—	—	—

Note. $\mathrm{mean}_{\pm \mathrm{SD}}$ across 3 seeds. Empty labels and Empty cand. are issue counts of two distinct attributes; Nonempty excludes issues with empty module labels. Within each benchmark, the best module F1 is bold and the second best underlined. Em dashes mark repeated counts and unreported LocBench scores.

with an initial reference ceiling of 1.0000. The advantage over Terminal-Prune remains on Verified, so the gain on that complete set does not depend on mixing issues of different initial retrieval difficulty.

Empty labels do not account for the main protocol gap either. After Table A.4 drops issues with empty module labels, the module-F1 gap of C3L over Terminal-Free on Verified moves from 12.48 to 12.44 points and stays in place. Candidate scale, representability, and the empty-label diagnostic agree on one conclusion: the initial candidates set the reachable level, and the gain of C3L is a better choice of which locations to keep from that shared start.

A.2 Label Cache Audit

From one SFT snapshot and one fixed random seed we build a frozen audit pool of 256 training issues by 8 trajectories. Terminal states that meet the cache trigger are deduplicated and validated, and rewards are recomputed on the original trajectories. The audit is not written into the official training cache. Define $\mathrm{AltPass} = |\mathcal{P}|/|\mathcal{V}|$, where $\mathcal{V}$ is every triggered trajectory and $\mathcal{P}$ is the subset

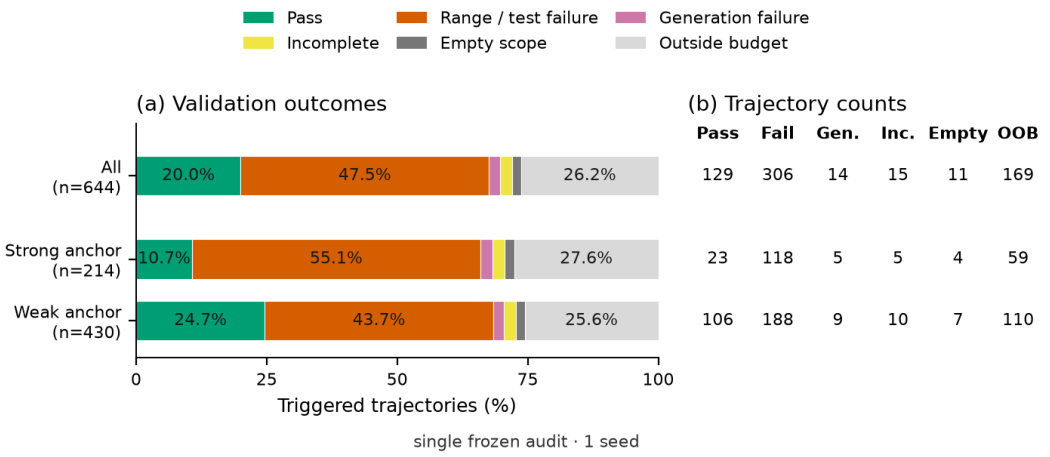


Figure A.3: Outcomes of execution-based label calibration. One frozen audit using 1 seed. Bars divide all triggered trajectories into six exclusive outcomes; the matrix gives counts. Fail, Gen., Inc., Empty and OOB denote range/test failure, generation failure, incomplete validation, empty scope and outside budget. All/strong/weak strata have 644/214/430 triggers, 435/141/294 completed validations, conditional pass rates 29.66%/16.31%/36.05%, 185/60/125 independent requests and 49/11/38 deduplicated labels. Request counts are not trajectory denominators; All is the union of the two strata.

Table A.5: Changes in scores, coverage ceilings and learning weights after label calibration.

Audit subset	**Δ R**	**Δ B_0**	**Δ B_t**	**Changed turns**	**Δ w**	**Coverage regained**
All triggered	0.1391	0.0213	0.1606	257	+0.0224	129
Validated paths	0.6944	0.1063	0.8016	257	+0.1117	129

Note. One frozen audit, using 1 seed. All triggered and Validated paths contain 644 and 129 trajectories; the latter are validated alternative-path trajectories. Changed turns are turns whose coverage reward changes. Coverage regained is the number of trajectories that regain complete acceptable file coverage. Δw is the mean change in w. Differences compare the same trajectories before and after expanding acceptable labels; B_0 and B_t use the respective label sets.

that passes the scope check and F2P/P2P and whose actual patch label differs from the reference label. The denominator includes empty scopes, failures, incomplete runs, and trajectories left uncalled because they fell outside the budget. Independent validation requests and deduplicated labels are counted separately.

Figure A.3 is direct evidence that a repair can succeed when the reference files are incomplete. Of 2,048 trajectories, 644 trigger. The 185 independent requests correspond to 129 alternative paths that pass the tests, so AltPass is 20.03%. On the 435 trajectories whose validation finished, the pass rate is 29.66%. These paths produce 49 deduplicated alternative labels. After calibration, 129 trajectories regain complete acceptable file coverage. In Table A.5 their mean terminal score rises by 0.6944 and their terminal ceiling rises by 0.8016. Expanding the label set cannot lower the maximum matched score or the ceiling, but the turn-level potential differences and the learning weights still have to be recomputed. The coverage reward changes on 257 turns, so calibration changes the credit assignment.

The cached labels also hold up under manual inspection. Two authors inspected 100 alternative labels sampled uniformly from the final cache of one seed: 84 are genuine repairs at a different location, 9 are valid repairs that also touch unnecessary locations, and 7 pass only because the tests are weak.

During official training the mean number of acceptable labels grows from 1.28 to 1.57, and the trigger rate falls from 31.4% to 21.2% (Table A.6). Only 14 issues reach the cap of four labels at the

Table A.6: Label-cache growth and calibration cost over training iterations.

Iteration	Trigger rate (%)	Mean labels	Issues at cap	Cal. GPU-h	Test CPU-h
1	31.4	1.28	6	6.3	79.6
2	24.7	1.46	11	4.9	62.6
3	21.2	1.57	14	4.2	53.8

Note. Mean labels is the mean number of acceptable labels. Issues at cap counts issues that have reached the cap of 4 acceptable label sets. Cal. GPU-h is calibration GPU-hours, which sum to 15.4; test CPU-hours sum to 196.0.

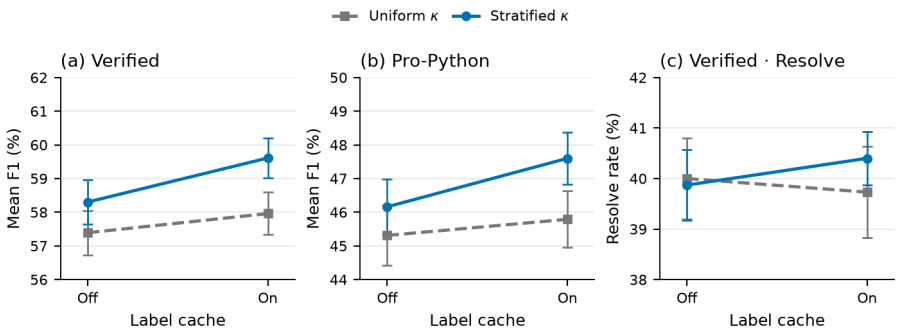


Figure A.4: Coverage-penalty stratification and label caching. Cache on/off crossed with uniform/stratified coverage coefficients; $\mathrm{mean}_{\pm\mathrm{SD}}$ across 3 seeds. F1 and resolve use separate scales; interaction increments are point estimates. Stratified core/full configurations are shared with Tables 1–3.

third iteration, so the cache stays sparse. The three iterations spend 15.4 GPU-hours on calibration generation and 196.0 CPU-hours on tests.

The audit and the held-out test show the same stratification. AltPass is 24.65% on weak-anchor trajectories and 10.75% on strong-anchor trajectories: when the reference cue is weaker, an execution check more often finds another valid repair path. Figure A.4 crosses the cache with stratification of the coverage penalty. Under stratified κ, the cache adds 1.31 points of Verified F1, and under a uniform coefficient it adds 0.57, a difference of differences of 0.73 points of F1 and 0.80 points of resolve rate. With the cache on, stratified κ beats the uniform coefficient by 1.64 points of Verified F1 (95% CI [0.04, 3.27]). On the held-out Verified test, Table A.7 reports weak-anchor gains of 2.04 points in reference F1 and 1.06 points in resolve rate, both larger than the strong-anchor gains of 0.57 and 0.00, although the paired resolve intervals cross zero. Together with the audit, the result supports assigning the coverage penalty by anchor strength and using execution checks to supply valid repair paths that the reference patch does not cover.

A.3 Repairer Transfer

We freeze the localization outputs and the repair procedure, and we replace the downstream repairer, Qwen3-Coder-30B, with Qwen2.5-Coder-32B, the model used for label calibration during training. Besides the absolute resolve rates, we test whether the cache increment of full C3L over C3L-core grows on the calibration repairer.

The localization gain transfers to the second repairer. In Table A.8, 4B C3L reaches 35.60% resolve with Qwen2.5-Coder-32B, above A^{2}Agent-Matched at 31.93%. At 8B the corresponding rates are 37.53% and 33.00%. Absolute resolve rates differ between the two repairers, and C3L leads at both locator scales. The effect of localization quality is not confined to the repair backend used in the main text.

Figure A.5 separates the process mechanism from the cache increment. At 4B the cache increment is 0.53 points on both repairers. At 8B it is 0.27 points on the repairer that took no part in calibration and 0.13 points on the calibration repairer. Every paired interval crosses zero. The present results

Table A.7: Cache increments on independent Verified test strata.

Stratum	Comparison	Issues	Δ Ref. F1 (pp)	Δ Resolve (pp)	95% CI
All	C3L vs C3L-core	500	+1.31	+0.53	[-0.93, 2.00]
Strong anchors	C3L vs C3L-core	249	+0.57	+0.00	[-1.87, 2.01]
Weak anchors	C3L vs C3L-core	251	+2.04	+1.06	[-1.06, 3.32]

Note. Differences are full C3L minus C3L-core under fixed reference scoring. Δ Ref. F1 is the reference-F1 difference. The 95% CI is the issue-paired interval for Δ resolve, using 3 locator seeds.

Table A.8: One-shot repair with two frozen repairers on Verified.

	Resolve (%)		
Method	**Qwen3-30B**	**Qwen2.5-32B**	**Diff. (pp)**
4B Terminal-Prune	36.53	30.60	-5.93
4B A^2Agent-Matched	37.33	31.93	-5.40
4B CodeScout-Matched	34.20	29.33	-4.87
4B FuseSearch-Seq	36.73	31.20	-5.53
4B C3L-core	<u>39.87</u>	<u>35.07</u>	-4.80
4B C3L	**40.40**	**35.60**	-4.80
8B Terminal-Prune	36.47	31.20	-5.27
8B A^2Agent-Matched	37.07	33.00	-4.07
8B C3L-core	<u>41.53</u>	<u>37.40</u>	-4.13
8B C3L	**41.80**	**37.53**	-4.27
Reference-locations	46.20	45.60	-0.60

Note. Qwen3-30B is Qwen3-Coder-30B-A3B, independent of training-time calibration; Qwen2.5-32B is the calibration repairer. Diff. is second minus first. Within each locator block, the best resolve rate is bold and the second best underlined. Trainable locator rows report means across 3 seeds; the reference row is deterministic. Paired cache increments appear in Figure A.5.

show no extra amplification on the calibration repairer, and the core already carries the main gain. The transfer check and the execution audit of the cache complement each other. Alternative labels correct the supervision of acceptable locations, and another repair backend can use the improvement in localization.

A.4 Optimization Objectives and Baseline Fidelity

We replace the update with GRPO, using within-group normalized advantages, importance-ratio clipping, a KL coefficient of 0.01, and the 12-round on-policy schedule of the GRPO baselines (§ 4.1). Rewards, grouping, candidate weights, and suffix decay stay fixed. The control runs with the cache off and with the cache on, and it includes a Terminal-Prune baseline trained under the same objective, which separates the gain of the process signals from the sensitivity of the cache to the form of the update.

The process signals are effective under both objectives. In Table A.9, the Verified Mean F1 gain of C3L-core over Terminal-Prune is 10.50 points under the imitation-weighted objective and 9.34 points under GRPO. Both paired intervals are positive, so the gain from contraction feedback and from state comparison holds across the update. The cache increment does not hold in the same way. In Figure A.6, imitation weighting raises F1 by 1.31 points and lowers PER by 1.79 points, while GRPO changes F1 by -0.23 and raises PER by 0.79 points. The difference of the F1 increments is 1.54 points. The paired intervals of the two cache increments are [-0.09, 2.70] and [-1.94, 1.48], and the interaction is reported as a point estimate. The direction matches the reason for a positive-weight objective: when the label set grows, a trajectory that scored lower keeps a chance to be imitated, and a later validated alternative path can receive credit.

Published numbers check the scale of the re-run. In Table A.10, all seven localization metrics of A^2Agent-Matched sit 1.20–1.55 points below the published values, and downstream resolve sits 0.74 points below, with no single metric as an outlier. The shared candidates and label parser, the

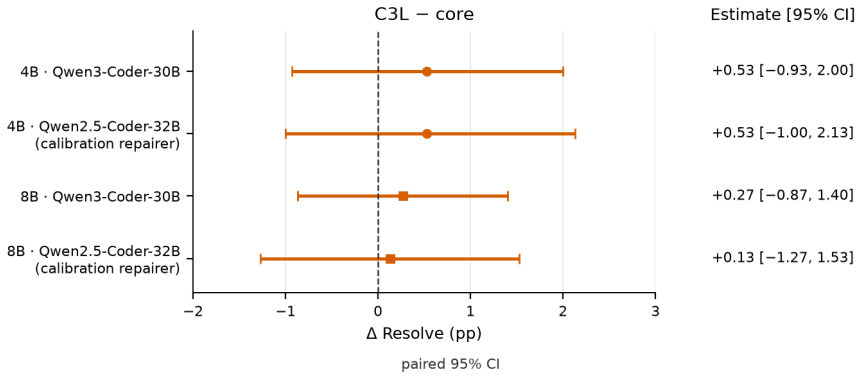


Figure A.5: Cache increments across repairers. Full C3L minus C3L-core in resolve rate, with issue-paired 95% CIs using 3 locator seeds. Circles/squares denote 4B/8B locators. Qwen2.5-Coder-32B is the calibration repairer; Qwen3-Coder-30B is independent of calibration. Absolute rates appear in Table A.8.

Table A.9: Process-signal gains under two optimization objectives.

Comparison	**Δ Verified Mean F1 (pp)**	**95% CI**
C3L-core - Terminal-Prune (imitation-weighted)	+10.50	[8.87, 12.16]
C3L-core - Terminal-Prune (GRPO)	+9.34	[7.54, 11.13]

Note. Issue-paired 95% CIs on Verified, using 3 seeds. Each comparison uses a Terminal-Prune baseline trained with the same objective as C3L-core.

three-seed aggregate, the tools, and the summary prompt all differ from the published setup. The comparison is one of scale. The main table uses the unified re-run. The reference-location input in Table A.11 resolves 46.20%, close to the published oracle of 45.93%, and supplies an independent reference for the repair backend. Published CodeScout uses a smaller turn budget and a different training scale. Published FuseSearch uses a parallel protocol and evaluates on a representable subset, both different from this paper, so those published scores do not enter the main table. The objective cross and these references show that the main process gain holds across objectives, and that the baseline comparison rests on an explicit adaptation.

A.5 Design Choices

This section takes C3L-core as the base and changes one design at a time among the reward, the grouping, and the decay, with reference labels held fixed, using 2 seeds. Protocol constraints use 3 seeds, with a terminal-training control and a full-C3L control matched to the condition each constraint acts on. Reward timing is checked on frozen trajectories and across three training iterations.

A.5.1 Reward composition and strength

Retrieval-weighted volume makes contraction more selective. In Figure A.7, replacing the default weights by a uniform count inside each granularity lowers Verified Mean F1 from 58.31% to 55.22% and raises PER from 12.96% to 18.74%. Table A.12 shows that the uniform count ends at a smaller volume (0.92% against 0.96%, both evaluated with the default weights) and at lower file recall. A smaller set is not by itself a high-quality localization. The reward has to distinguish the initial relevance of the candidates.

The one-factor scan in Figure A.8 shows the balance between contraction and coverage. Raising α to 0.2 lifts the empty-prediction rate from the default 0.20% to 2.30%. Doubling κ lowers PER to 10.92% and raises the P90 of the terminal file count to 3.0. Stronger compression and stronger retention each move away from the default, and neither exceeds the default Mean F1. The reported scan of the weight range also supports the default $[0.2, 5]$. Parameter choice is tied here to how the

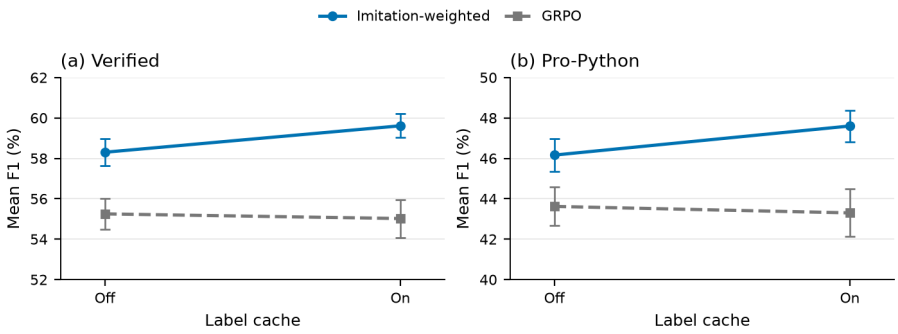


Figure A.6: Optimization objective and label-cache interaction. Cache Off/On denotes C3L-core/full C3L; $\text{mean}_{\pm\text{SD}}$ across 3 seeds. Verified cache increments under imitation-weighted/GRPO objectives are +1.31/-0.23 F1 pp and -1.79/+0.79 PER pp; differences of increments are +1.54 and -2.58 pp, reported as point estimates. Full-method PER is $11.49_{\pm 0.54}\%$/$19.06_{\pm 0.71}\%$, terminal volume is $0.98_{\pm 0.01}\%$/$0.93_{\pm 0.02}\%$ of initial, and training cost is 185.7/193.4 GPU-h, respectively.

Table A.10: A^{2}Agent-Matched re-runs and published A^{2}Agent results.

Metric	Backbone	Published	Re-run	Difference
Verified File F1 (%)	Qwen3-4B-Instruct	72.64	71.33	-1.31
Verified Module F1 (%)	Qwen3-4B-Instruct	46.94	45.65	-1.29
Verified Function F1 (%)	Qwen3-4B-Instruct	38.59	37.22	-1.37
Verified Mean F1 (%)	Qwen3-4B-Instruct	52.72	51.40	-1.32
Verified Mean F1 (%)	Qwen3-8B	54.10	52.90	-1.20
Pro-Python Mean F1 (%)	Qwen3-4B-Instruct	41.93	40.38	-1.55
Pro-Python Mean F1 (%)	Qwen3-8B	43.24	41.80	-1.44
Verified Resolved (%)	Qwen3-4B-Instruct	38.07	37.33	-0.74

Note. Published values are single-run reports; re-run values are means across 3 seeds. Difference is re-run minus published. Re-runs use the shared candidate universe, label parser, tools and prompts.

candidate set actually moves, and the comparison is not a ranking of configurations by one terminal score.

A.5.2 Survivor signatures and advantage fusion

Effective group coverage (EGC) is the fraction of decision turns whose group contains at least two independent rollouts and whose return standard deviation exceeds ε_g. The denominator includes degenerate groups and single-rollout groups. The frozen-state diagnostic also reports the within-group standard deviation of file coverage, which measures how similar the residual tasks under comparison are.

What the groups compare, not how many groups exist, decides the value of the turn-level advantage. Issue-only, history ($H = 2$), and default signature groups all reach near-saturated EGC (100.00%, 99.97%, and 99.83%; Table A.13), yet their Verified F1 is 55.77%, 56.91%, and 58.31% (Figure A.9), against 55.52% for trajectory-only advantages (Figure A.10). Among these keys the survivor signature also forms the most homogeneous groups, with within-group coverage SD 0.0042 against 0.0152 for history groups and 0.0297 for issue groups. A size-matched random signature (SD 0.0288, 56.11%) shows that the gain comes from the state the key encodes rather than from group size. Matching the full survivor set goes too far: it drives the SD to 0 but leaves 28.65% of decision turns without a valid group and reaches 55.38%. The volume bin carries the signature: the top-level directory alone rarely changes and reaches 55.73%, and adding the bin lifts F1 to 58.31%. Paired over issues, the signature beats issue groups by 2.54 points (95% CI [1.17, 3.91]). For the history key we add a third seed and compare it with the main three-seed C3L-core, which uses the

Table A.11: Reference-location repair as an independent backend comparison.

Comparison	Published A^{2}Agent (%)	This work (%)
Oracle/Reference-locations resolved	45.93	46.20

Note. Both results use Agentless with Qwen3-Coder-30B-A3B-Instruct on Verified. These are reference-input success rates, not learned-locator scores. This work is an independent reference for the Agentless backend and context packer.

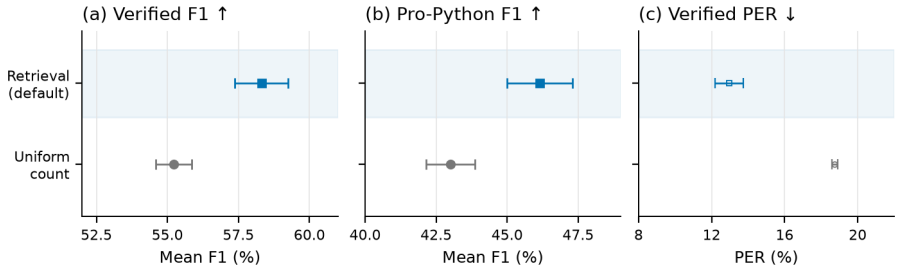


Figure A.7: Retrieval-weighted versus uniform-count volume. C3L-core with fixed α=0.1, κ multiplier=1 and weight range [0.2,5]. F1 and Verified PER show $\text{mean}_{\pm\text{SD}}$ across 2 seeds. Candidate-state diagnostics appear in Table A.12.

default signature: the signature leads by 1.35 points on Verified ([0.19, 2.51]; history $56.95_{\pm 0.46}$) and 1.19 on Pro-Python (history $44.97_{\pm 0.60}$), and lowers PER from 15.38% to 13.28% (Table B.6).

In Figure A.10, a trajectory-only advantage and a turn-only advantage reach Verified F1 of 55.52% and 54.87%, both below equal-weight fusion at 58.31%. The trajectory comparison supplies a reference for the outcome of the whole rollout. The turn comparison separates decisions that face similar residual tasks. The two complement each other. With the turn advantage alone, zeroing a degenerate group drops the relative ranking, while the positive imitation weight remains. Equal-weight fusion lets those turns use the trajectory-level difference as well.

A.5.3 Normalized continuous suffix decay

To separate a retrieval miss from a mistaken exclusion during search, we compare normalized continuous decay, unnormalized continuous decay, no decay, and a hard cutoff after the first coverage loss. A partial-coverage suffix satisfies $0 < B_x^f(S_{t-1}) < B_x^f(S_0)$. SuffixTok is, among trajectories that incur a coverage loss, the fraction of that batch's agent tokens which lie after the turn of the first loss. The change in F1 after the loss likewise excludes the action that caused the loss.

Normalization keeps an initial retrieval gap from downweighting the whole trajectory. In Table A.14, the 33 issues with incomplete initial file coverage have first-turn mean d of 0.5657 under the unnormalized rule and 1.0000 once normalized. The corresponding Verified F1 in Figure A.11 rises from 57.20% to 58.31%. Hard cutoff scores lower, at 54.91%. Under the default rule, partial-coverage trajectories still gain 21.00 points of state F1 on average after the loss, so those later actions remain worth learning.

Figure A.12 isolates the decay rule on one frozen pool. The default rule keeps mean $d = 0.7026$ and $w = 0.5199$ on partial-coverage suffixes. Hard cutoff sets both to zero. On zero-coverage suffixes the default mean weight has already fallen to 0.0036. This diagnostic and the quality comparison from independent training both support continuous decay: the learning weight stays while a target can still be improved, and later credit falls as reachable coverage is lost.

A.5.4 Output binding and irreversible contraction

The output-binding control, trained with a terminal reward, allows an independent submission $\hat{y} \subseteq S_T$. The irreversibility control, under full C3L, allows `restore` to put a deleted entity back into

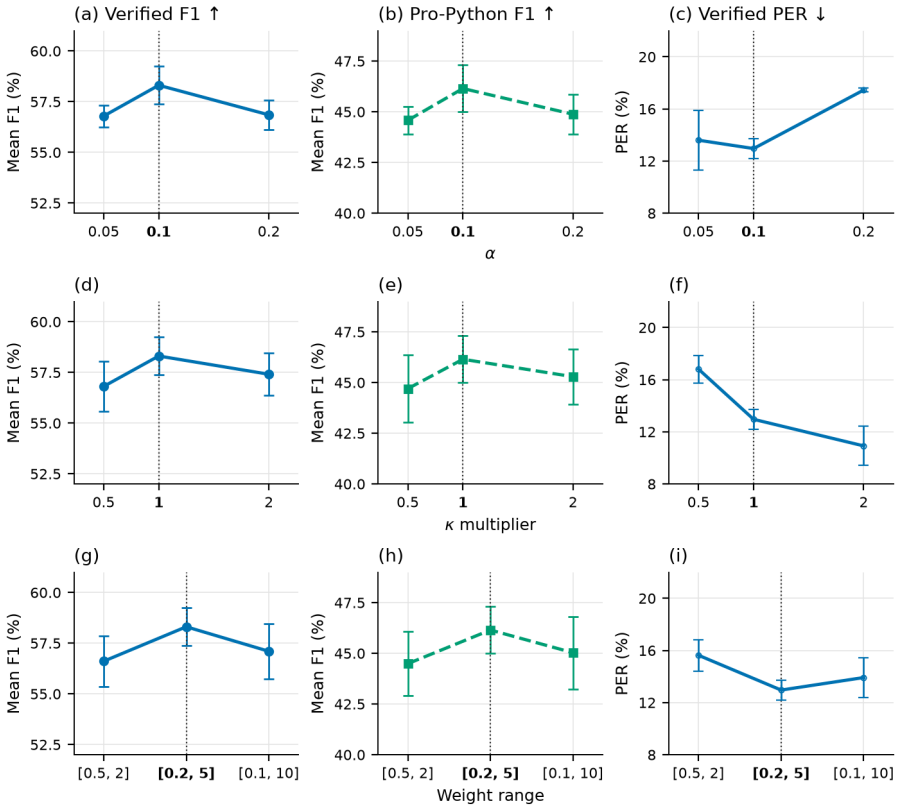


Figure A.8: Sensitivity to reward strength and learning-weight range. One-factor scans of α, κ multiplier and weight range; other C3L-core parameters remain at their defaults. $\text{mean}_{\pm\text{SD}}$ across 2 seeds; PER is for Verified. Dotted lines mark defaults and lines connect only evaluated settings. Candidate-state diagnostics appear in Table A.12.

Table A.12: Candidate-state diagnostics for reward and weight scans.

				Verified state			
Weighting	α	κ	**Range**	**Recall (%)**	**Volume (%)**	**Empty (%)**	**Files P90**
Retrieval-weighted (default)	0.1	1	[0.2,5]	$85.47_{\pm 0.66}$	$0.96_{\pm 0.01}$	$0.20_{\pm 0.28}$	$2.0_{\pm 0.0}$
Uniform count	0.1	1	[0.2,5]	$80.93_{\pm 0.34}$	$0.92_{\pm 0.00}$	$0.30_{\pm 0.14}$	$2.0_{\pm 0.0}$
Retrieval-weighted	0.05	1	[0.2,5]	$85.42_{\pm 2.14}$	$0.99_{\pm 0.01}$	$0.30_{\pm 0.42}$	$2.5_{\pm 0.7}$
Retrieval-weighted	0.2	1	[0.2,5]	$81.20_{\pm 0.57}$	$0.89_{\pm 0.01}$	$2.30_{\pm 0.42}$	$2.0_{\pm 0.0}$
Retrieval-weighted	0.1	0.5	[0.2,5]	$82.09_{\pm 0.84}$	$0.91_{\pm 0.01}$	$0.40_{\pm 0.28}$	$2.0_{\pm 0.0}$
Retrieval-weighted	0.1	2	[0.2,5]	$86.93_{\pm 1.51}$	$1.00_{\pm 0.02}$	$0.10_{\pm 0.14}$	$3.0_{\pm 0.0}$
Retrieval-weighted	0.1	1	[0.5,2]	$83.03_{\pm 0.77}$	$0.94_{\pm 0.01}$	$0.80_{\pm 0.57}$	$2.0_{\pm 0.0}$
Retrieval-weighted	0.1	1	[0.1,10]	$84.64_{\pm 1.19}$	$0.97_{\pm 0.01}$	$0.90_{\pm 0.14}$	$2.1_{\pm 0.1}$

Note. $\text{mean}_{\pm\text{SD}}$ across 2 seeds, on Verified. Blocks vary weighting, α, the κ multiplier, and the learning-weight range. κ is that multiplier; Volume is the terminal volume as a percent of the initial volume; Empty is the empty-prediction rate; Files P90 is the terminal file-count percentile. All terminal volumes are evaluated with the same default retrieval weights. Quality and PER appear in Figures A.7–A.8.

the candidates. Restoration reuses the original reward, so it can produce a positive difference in the coverage potential. Recovery success takes as its denominator the trajectories that first lose reference-file coverage, and asks whether that file is restored before termination. Without restoration the rate is undefined.

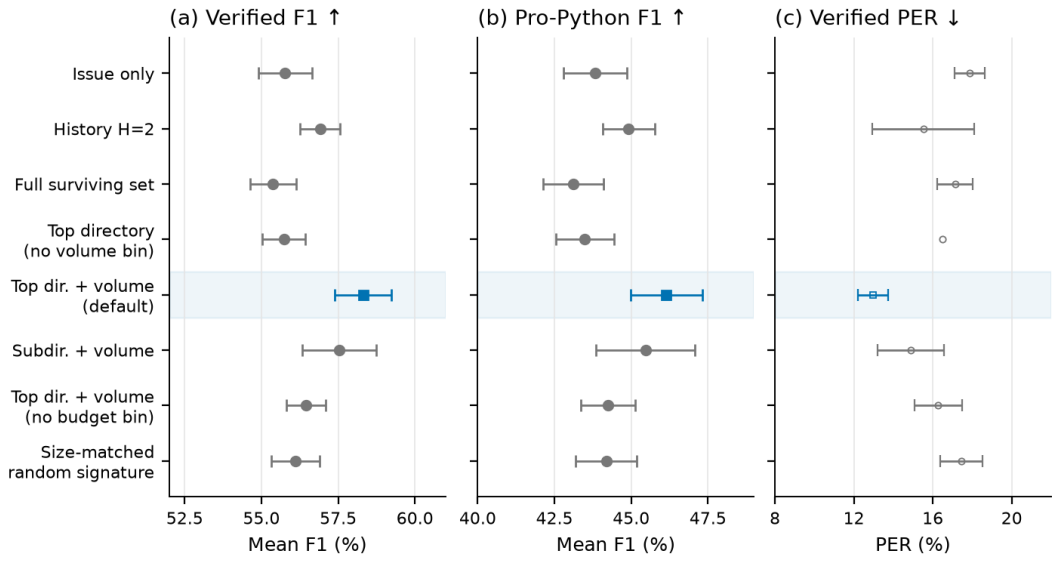


Figure A.9: Survivor-signature grouping ablations. $\text{mean}_{\pm\text{SD}}$ across 2 seeds for F1 and Verified PER. The default combines top-level directories, volume and remaining-budget bins. Top directory without a volume bin has reported PER SD 0.00. Frozen-state diagnostics are given in Table A.13; the three-seed comparison with the history key appears in Table B.6.

Table A.13: Frozen-state diagnostics for survivor-signature grouping.

Grouping key	**Budget**	**EGC (%)**	**Rollouts**	**Cov. SD**
Issue only	Yes	100.00	8.00	0.0297
History H=2	Yes	99.97	7.96	0.0152
Full survivor set	Yes	71.35	4.99	0.0000
Top directory, no volume	Yes	99.92	7.90	0.0069
Top dir. + volume (default)	Yes	99.83	7.87	0.0042
Subdir. + volume	Yes	97.31	6.64	0.0011
Top dir. + volume, no budget	No	99.91	7.89	0.0044
Size-matched random signature	Yes	99.85	7.03	0.0288

Note. Point estimates from frozen decision states. EGC is the share of decision turns in valid comparison groups. Budget is the remaining-budget bin. Rollouts is the mean number of distinct rollouts in valid groups; because a group can hold several turns of the same rollout, a finer key can split turns by contraction stage without removing rollouts, which is why Cov. SD falls much faster than Rollouts. Cov. SD is the within-group coverage standard deviation, a measure of state heterogeneity rather than F1 uncertainty. "No volume" omits the volume bin; "no budget" omits the remaining-budget bin. Retrained quality uses 2 seeds in Figure A.9.

In Figure A.13, allowing an independent subset submission moves the output F1 of terminal-only training from 47.80% to 48.15%, drops survivor-set F1 to 24.11%, and raises the mean number of surviving functions from 5.45 to 17.44. In Table A.15 the number of contraction decisions also falls from 11.4 to 6.8. An independent submission lets the model postpone the choice until the end. Strict binding requires the intermediate candidate state itself to carry the quality of the final answer.

Allowing restoration changes what the timing of the coverage signal means. On Verified the mean number of restorations is 3.42, and 31.60% of trajectories that excluded a reference file recover it, yet terminal PER still rises to 16.83% and Mean F1 falls from 59.61% to 54.86%. On Pro-Python, quality falls from 47.60% to 42.55% (Figure A.14). On 14.27% of Verified turns the ceiling rises, so a coverage deduction becomes a temporary loss that a later turn can return. Relaxing output binding on top of restoration lowers quality further. The two constraints do different work. Binding lands

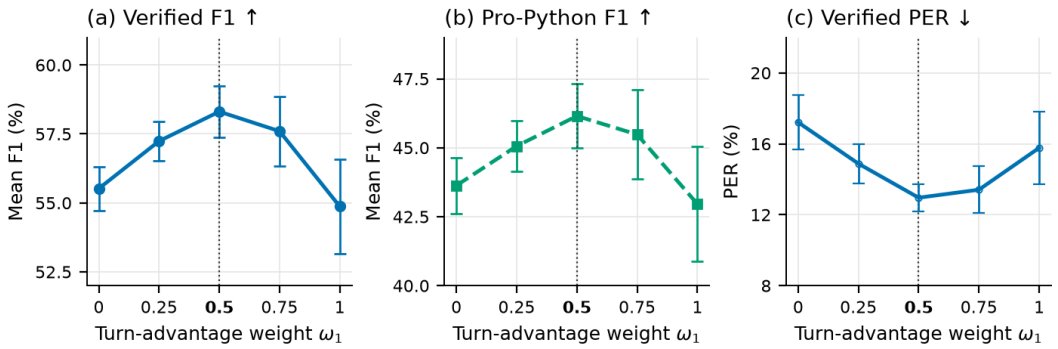


Figure A.10: Fusion of trajectory and turn advantages. Scan of the turn-advantage weight at the default grouping signature; $\text{mean}_{\pm\text{SD}}$ across 2 seeds. PER is for Verified. The default gives equal weight to both components; endpoints use trajectory-only and turn-only comparisons.

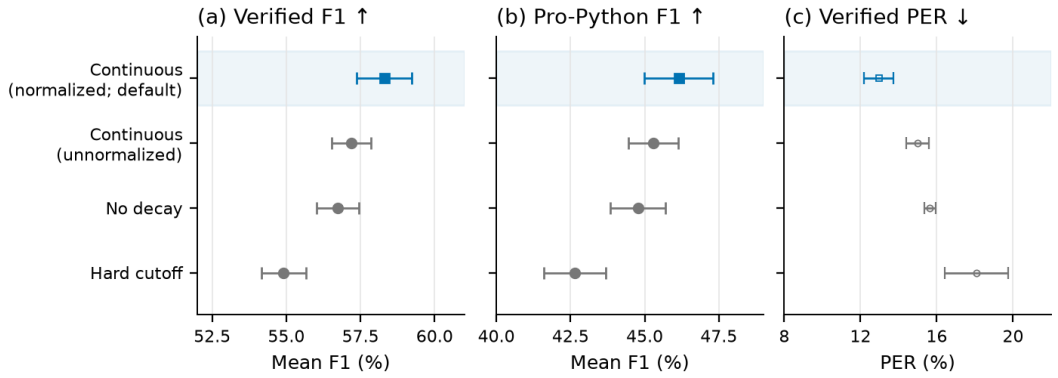


Figure A.11: Comparison of suffix-decay rules. F1 and Verified PER show $\text{mean}_{\pm\text{SD}}$ across 2 seeds from independently trained variants. Initial and post-loss diagnostics appear in Table A.14; shared frozen-pool weights appear in Figure A.12.

the decision on the survivor state. Irreversibility makes the coverage loss equal to the future choices the current action has consumed.

A.5.5 Reward timing and training dynamics

Table A.16 checks the equal-return control of § 4.5. With trajectory advantages alone, the difference in total return between the two reward timings is exact zero, and the residuals in per-token weight and in loss sit at the scale of FP32 accumulation. Once the turn advantage is fused in, the total return is still identical, and the weights and the loss change by a material amount. On the same 267-issue SWE-Smith validation split, Figure A.15 shows that turn-level reward with the fused advantage leads terminal delivery after each of the three iterations. After the third iteration, turn-level delivery is at 64.50% and terminal delivery at 62.30%, so the advantage of timely feedback builds up over the course of training.

A.6 Generalization and Task Analysis

We reuse the per-issue results from the main text, with no additional training, and stratify by repository scale, by the number of files in the reference edit, by anchor strength, by the density of highly

Table A.14: Initial and post-loss diagnostics for suffix decay.

Decay rule	**Initial d**	**Partial cov.**	**ΔF1 (pp)**	**SuffixTok (%)**
Continuous (normalized; default)	1.0000	21	+21.00	40.57
Continuous (unnormalized)	0.5657	22	+23.10	40.97
No decay	1.0000	21	+19.62	41.86
Hard cutoff	1.0000	31	+26.12	38.26

Note. Diagnostics use each independently trained variant's own trajectories; retrained quality uses 2 seeds. Initial d is the mean over 33 issues with incomplete initial file coverage. Partial cov. is the number of partial-coverage trajectories. ΔF1 is the subsequent change, and both it and SuffixTok exclude the first loss-of-coverage turn. Shared-pool weight comparisons appear in Figure A.12.

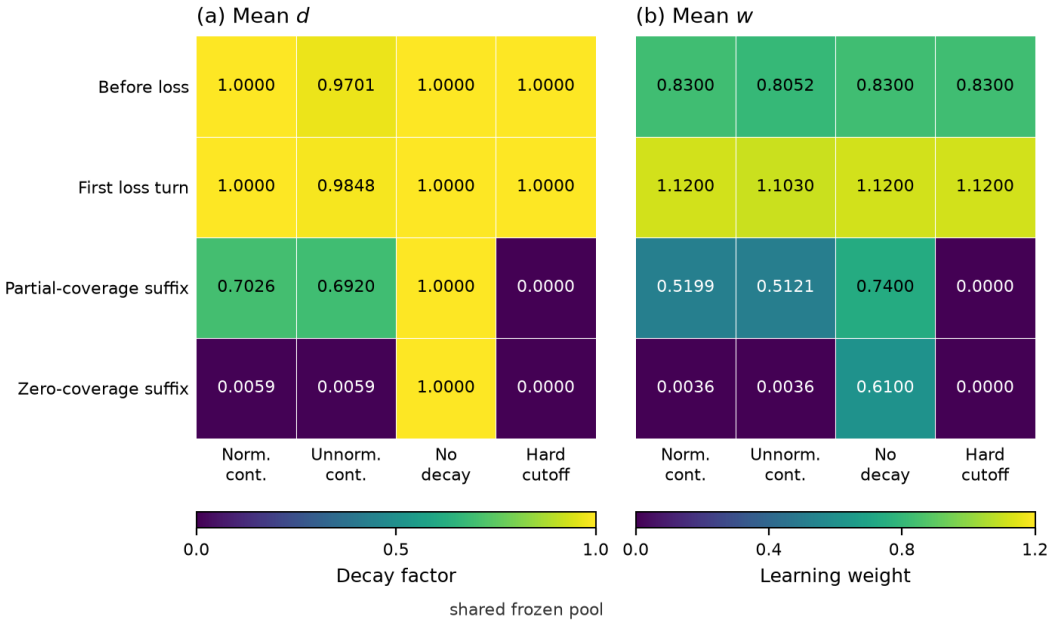


Figure A.12: Suffix factors and learning weights on shared trajectories. Four decay rules evaluated on the same frozen trajectory pool. Panels report mean d and w by turn stratum. Norm. cont. and Unnorm. cont. denote normalized and unnormalized continuous decay. Retrained quality appears in Figure A.11.

related distractors, and by the official LocBench issue type. Stratum thresholds are fixed from instance metadata. Distractor density $\mathrm{DD}(x)$ counts, among the top 50 fused-retrieval candidates, the entities that are absent from the reference label and that share a symbol-name prefix or a directory with some reference entity. The symbol prefix is the first segment of a snake_case or CamelCase split. DD is computed offline from the reference labels and does not enter the localization input. Repository scale and DD are tertiles within each benchmark. Every stratum reports its issue count and its initial reference ceiling.

The three-granularity breakdown shows that the quality gain of C3L is a gain in both precision and recall (Figure A.16, Figure A.17, and Figure A.18). On Verified, 4B C3L raises precision and recall over Terminal-Prune at every granularity. File precision rises from 65.07% to 78.06% and recall from 78.47% to 86.79%, while the mean predicted counts of files, modules, and functions fall from 1.58/2.81/5.45 to 1.46/2.58/4.83. A more compact set still retains more reference locations, in line with the localization and repair results in the main text. Macro F1 is computed per issue. The macro averages of precision and recall in the figures do not reconstruct that F1. Output size is a further diagnostic of how selectively the set is chosen.

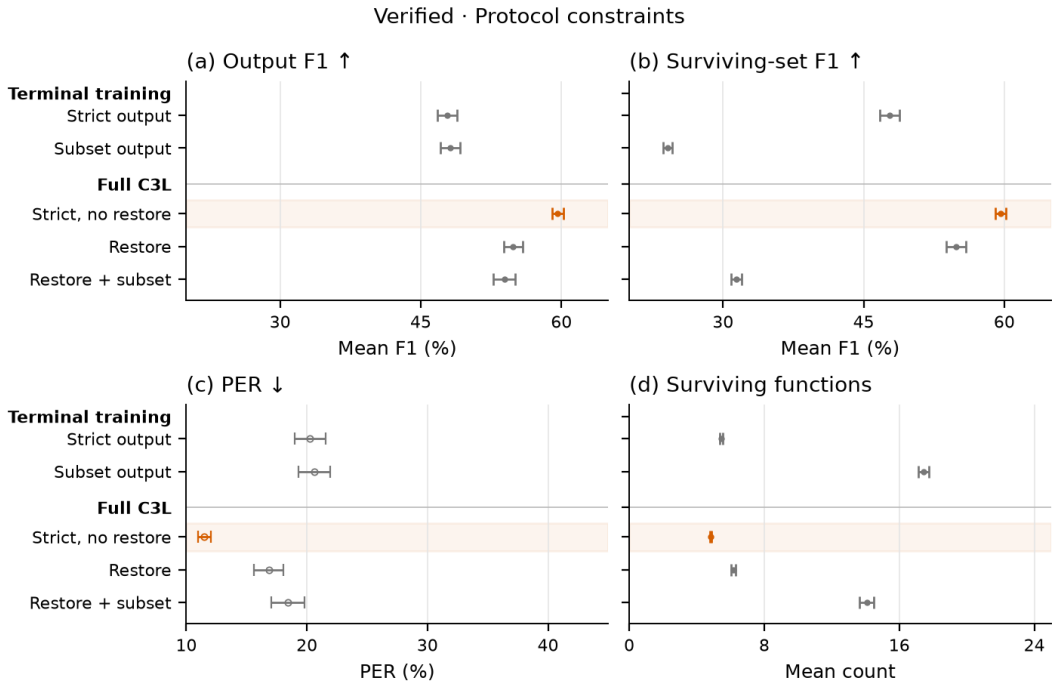


Figure A.13: Output binding and restoration on Verified. $\text{mean}_{\pm\text{SD}}$ across 3 seeds. The upper block compares strict versus subset output under terminal-only training; the lower block compares restoration options under full C3L. These are distinct training controls. Under strict binding, survivor-set F1 equals output F1 by definition. Action diagnostics appear in Table A.15.

Table A.15: Action diagnostics for output binding and restoration on Verified.

Configuration	Prune	Restore	Recovery (%)	UB (%)
Terminal training: Strict output	11.4	0.00	—	0.00
Terminal training: Subset output	6.8	0.00	—	0.00
Full C3L: Strict, no restore	12.9	0.00	—	0.00
Full C3L: Restore	15.7	$3.42_{\pm 0.19}$	$31.60_{\pm 2.44}$	$14.27_{\pm 0.81}$
Full C3L: Restore + subset	9.3	$2.15_{\pm 0.24}$	$28.94_{\pm 2.90}$	$9.63_{\pm 0.77}$

Note. $\text{mean}_{\pm\text{SD}}$ across 3 seeds for restoration statistics; zeros without restoration are structural. Prune and Restore are mean counts of pruning decisions and restore actions. Recovery is the recovery-success rate; it uses trajectories that lose reference-file coverage and is undefined without restoration. UB is the share of decision turns on which the reachable ceiling increases. Terminal-training and full-C3L rows are distinct controls.

The strata give the most direct support for contraction in the presence of highly related distractors. All 52 paired stratum comparisons in Figure A.19, Figure A.20, and Figure A.21 have positive intervals. On Verified, the gain of 4B C3L over Terminal-Prune rises across the low, middle, and high DD tertiles, from 8.70 to 11.70 to 15.05 points. At 8B the gains are 8.57, 11.84, and 15.58. Both scales on Pro-Python increase in the same direction. Cramér's V between DD and anchor strength is 0.11, so the two axes describe different attributes of an issue. The gain holds across issue types. On LocBench the 4B Feature stratum gains 17.70 points, and the 8B Bug stratum gains 16.68. Larger gains where distractors are dense match a method that scores the exclusion itself.

Giving each repository equal weight checks whether large repositories dominate the aggregate. Figure A.22 shows the 12 Verified repositories. In Table A.17, repository-balanced F1 of 4B C3L is 56.31%, 47.15%, and 56.07% on the three benchmarks, above both Terminal-Prune and A^{2}Agent-

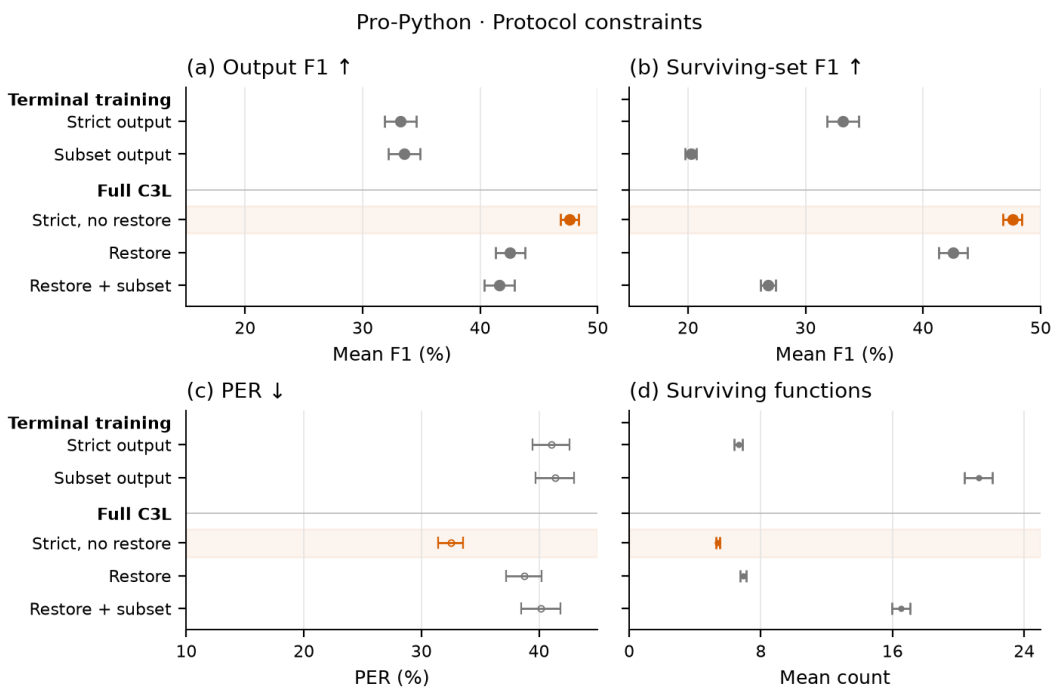


Figure A.14: Output binding and restoration on Pro-Python. $\text{mean}_{\pm\text{SD}}$ across 3 seeds. Terminal-training output-binding controls and full-C3L restoration controls use distinct baselines. Strict binding makes survivor-set F1 identical to output F1.

Table A.16: Numerical equivalence of reward timing on frozen trajectories.

Comparison	Maximum total-return difference	Maximum token-weight difference	Loss difference
Trajectory only: per-turn vs terminal	0.00e+00	5.96e-08	1.19e-07
Trajectory + turn: per-turn vs terminal	0.00e+00	3.14e-01	8.72e-02

Note. Both timings have identical total returns. Trajectory-only weight and loss residuals are at FP32 accumulation precision. Fused turn advantages change token weights and loss despite the identical total return.

Matched, and 8B keeps that order. Stratum axes can contain the same issues, and the per-repository point estimates carry no separate significance claim. Together they show that the overall gain covers different repositories and task attributes, and that it is larger when distractor candidates are denser.

A.7 Data Scaling and Compute Efficiency

We test scaling in data and in compute. The training pool grows from 2,400 issues to 10,400 (Table A.18). The 267-issue validation set and the shared SFT checkpoint stay fixed, and the added issues enter only the training that follows SFT. We compare three full iterations with a fixed rollout total, and we also limit which checkpoints can finish by cumulative GPU-hours. Pool expansion and equal-compute quality use 2 seeds. Training cost is the mean across 3 seeds.

The equal-compute comparison shows which stage of training the gain requires. Let $H = 185.7$ GPU-hours, the three-seed mean cost of full 4B C3L. Figure A.23 evaluates only checkpoints that finished inside each budget. At $H/3$, C3L still has only the SFT checkpoint, Verified F1 is 44.42%, and the paired difference from Terminal-Prune is -1.85 points (95% CI [-2.81, -0.88]). Once the

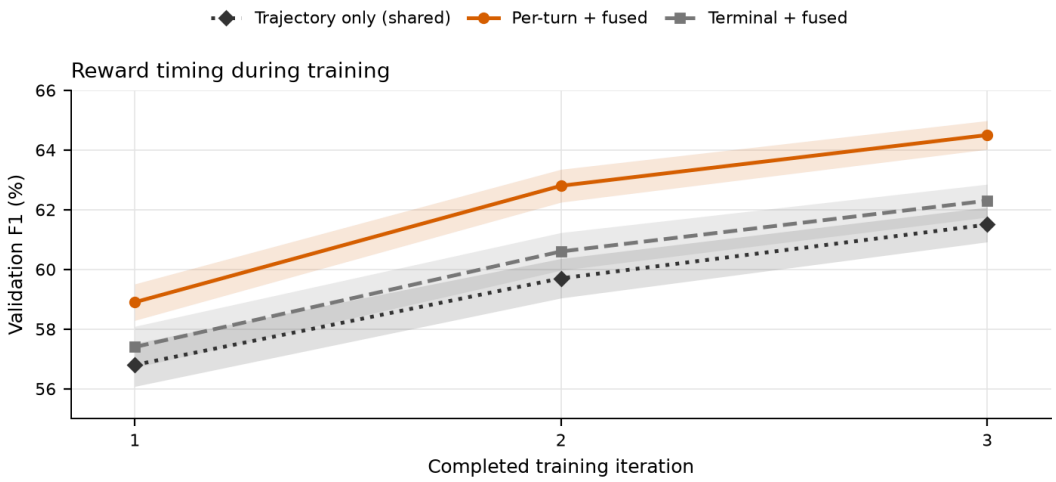


Figure A.15: Validation dynamics under different reward timings. $\text{mean}_{\pm\text{SD}}$ across 3 seeds after completed iterations 1–3 on the same 267-issue SWE-Smith validation split. This is in-distribution validation. Trajectory-only timings share one curve by telescoping; no iteration-zero point is introduced. Frozen-trajectory equivalence is checked in Table A.16.

Table A.17: Repository-balanced localization quality.

		Repository-balanced F1 (%)		
Benchmark	**Model**	**Terminal-Prune**	**A^{2}Agent**	**C3L**
Verified	4B	45.22	48.89	**56.31**
	8B	46.98	50.50	**57.63**
Pro-Python	4B	32.40	39.39	**47.15**
	8B	33.59	41.55	**48.39**
LocBench V1	4B	40.48	46.93	**56.07**
	8B	41.35	48.79	**57.19**

Note. First macro-average issues within each repository, then give repositories equal weight; these aggregates differ from the issue-macro results in Table 1. Within each row, the best F1 is bold and the second best is underlined. A^{2}Agent denotes A^{2}Agent-Matched.

budget reaches H, C3L is at 59.62% and Terminal-Prune at 47.81%, and the paired advantage is 11.81 points ([10.05, 13.60]). At $2H/3$, core leads the full method, 56.11% against 52.74%, because the cost of calibration changes how far training proceeds inside a limited budget. Figure A.24 splits that cost. Core uses 181.2 GPU-hours and Terminal-Prune uses 168.4. Full C3L uses 185.7, of which calibration generation is 15.4 GPU-hours, with tests counted separately at 196.0 CPU-hours. Its sampling and optimization take 166.1 GPU-hours, 10.9 fewer than C3L-core, consistent with its shorter trajectories (292.1K against 315.0K tokens per Verified issue at inference), so calibration adds only 4.5 GPU-hours net. A larger training pool brings a further gain (Figure A.25). Under three full iterations, C3L F1 on Verified, Pro-Python, and LocBench rises from 59.62%/47.58%/55.01% to 60.53%/49.21%/56.85%. In the matched-sample setting, each iteration draws 2,400 issues without replacement, stratified by repository, and draws again between iterations. With 57,600 rollouts per seed held fixed, the larger pool visits 5,666 distinct issues, and Verified and Pro-Python still rise by 0.47 and 0.86 points. This setting matches the number of sampled trajectories and supports the value of broader task coverage. Its compute cost is read separately from the equal-GPU-hour comparison above.

At inference, a larger model and parallel tools change different costs. In Table A.19, 8B C3L takes 342.7 seconds per Verified issue and 4B takes 265.2 seconds. The higher quality comes with a

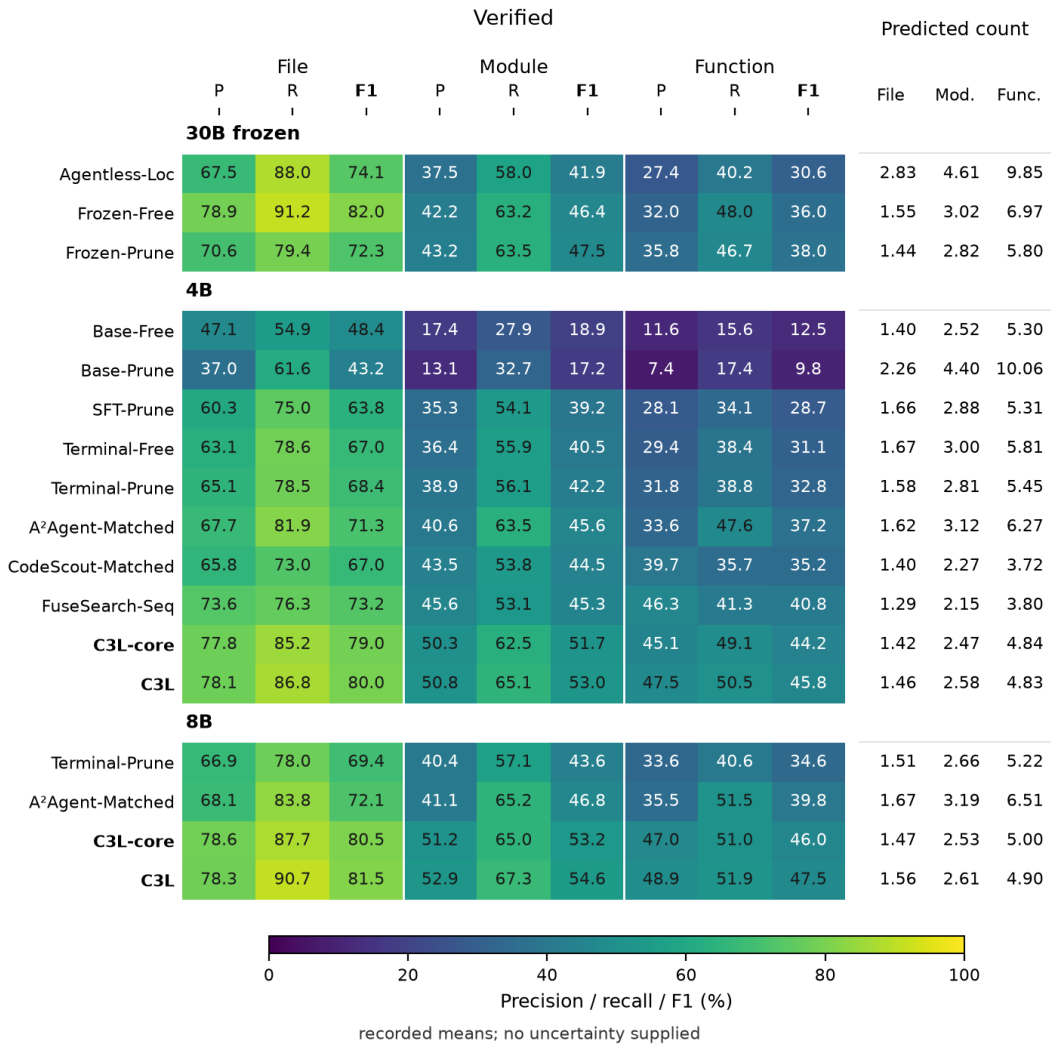


Figure A.16: Three-granularity localization on Verified. Macro precision, recall and F1, with predicted file/module/function counts, for 17 model/method configurations. Trainable rows aggregate 3 seeds; untrained/frozen rows are single runs. Cell labels round percentages to one decimal and counts to two decimals. Macro F1 is computed per issue, not reconstructed from macro precision and recall.

Table A.18: Training pools and full three-iteration rollout counts.

Training pool	**Training issues**	**Repositories**	**Rollouts per seed**
SWE-Smith-Python 2.4K	2,400	112	57,600
SWE-Smith-Python 10.4K (+8K)	10,400	117	249,600

Note. Both pools share the same 267-issue validation split and exclude test repositories. Rollout counts are per seed for three full iterations; the fixed-rollout comparison uses 57,600 trajectories per seed for either pool.

longer model forward pass. In Figure A.26, native parallel FuseSearch lowers the sequential run from 328.1 seconds to 126.2 seconds, and its cumulative token count remains 447.4K. Sequential C3L takes 265.2 seconds and 292.1K tokens at a higher F1 (59.61% against 52.65% for parallel FuseSearch). Parallelism shortens the wall-clock time of tool round-trips. Candidate contraction reduces how much text is read in total. The cost profile of C3L is then concrete: with sufficient

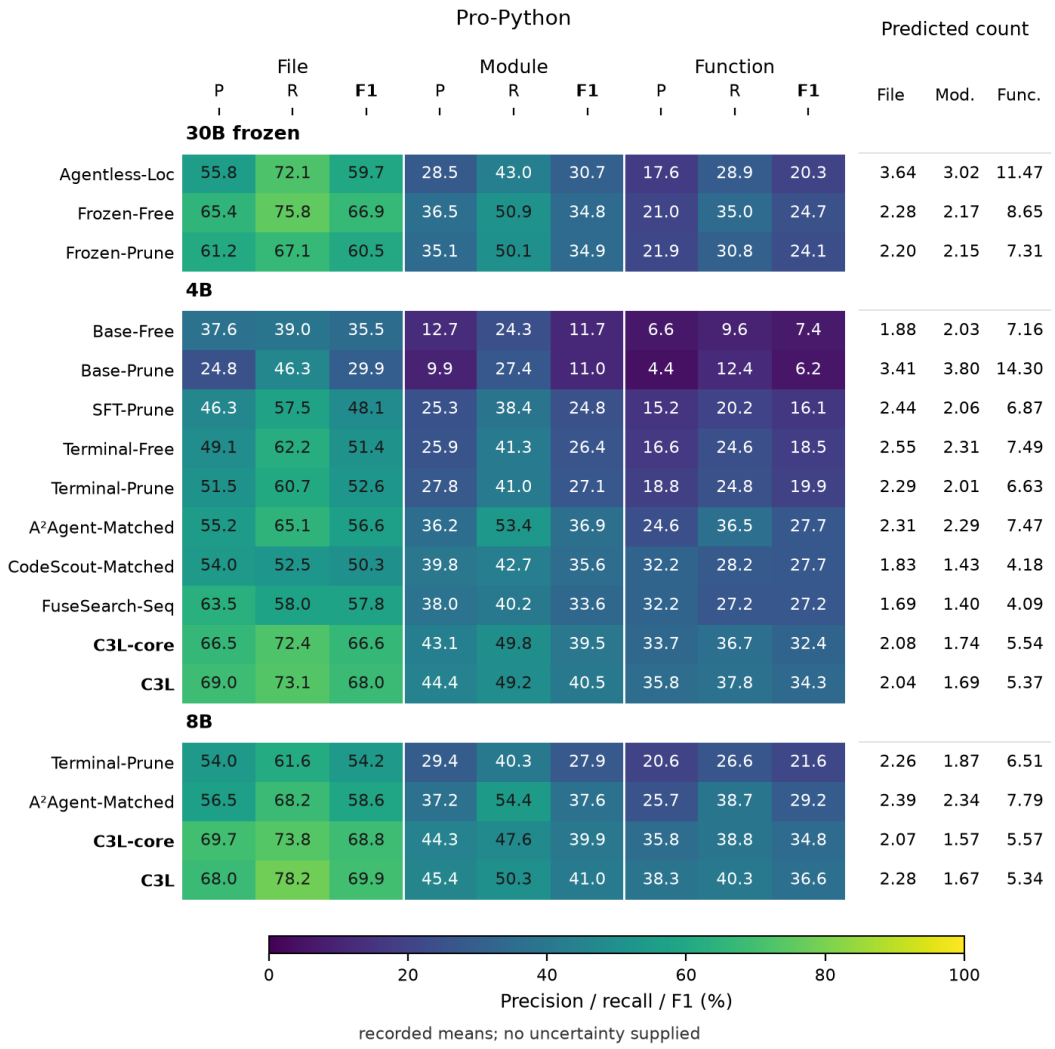


Figure A.17: Three-granularity localization on Pro-Python. Macro precision, recall and F1 and predicted set sizes for the same 17 configurations, formatted as in Figure A.16.

Table A.19: Inference cost of 4B and 8B C3L.

Model	Benchmark	Total tokens (K)	Localization latency (s)
4B	Verified	$292.1_{\pm 2.2}$	$265.2_{\pm 1.0}$
4B	Pro-Python	$305.6_{\pm 7.0}$	$263.8_{\pm 4.2}$
8B	Verified	$306.1_{\pm 10.5}$	$342.7_{\pm 8.0}$
8B	Pro-Python	$324.2_{\pm 11.0}$	$344.8_{\pm 8.0}$

Note. $\text{mean}_{\pm \text{SD}}$ across 3 seeds under the 20-call budget. Total tokens include cumulative input and output; latency is measured in seconds per issue.

training it keeps a quality lead at equal compute, a larger pool continues to help, and the reading saved at inference is a different axis from parallel speedup.

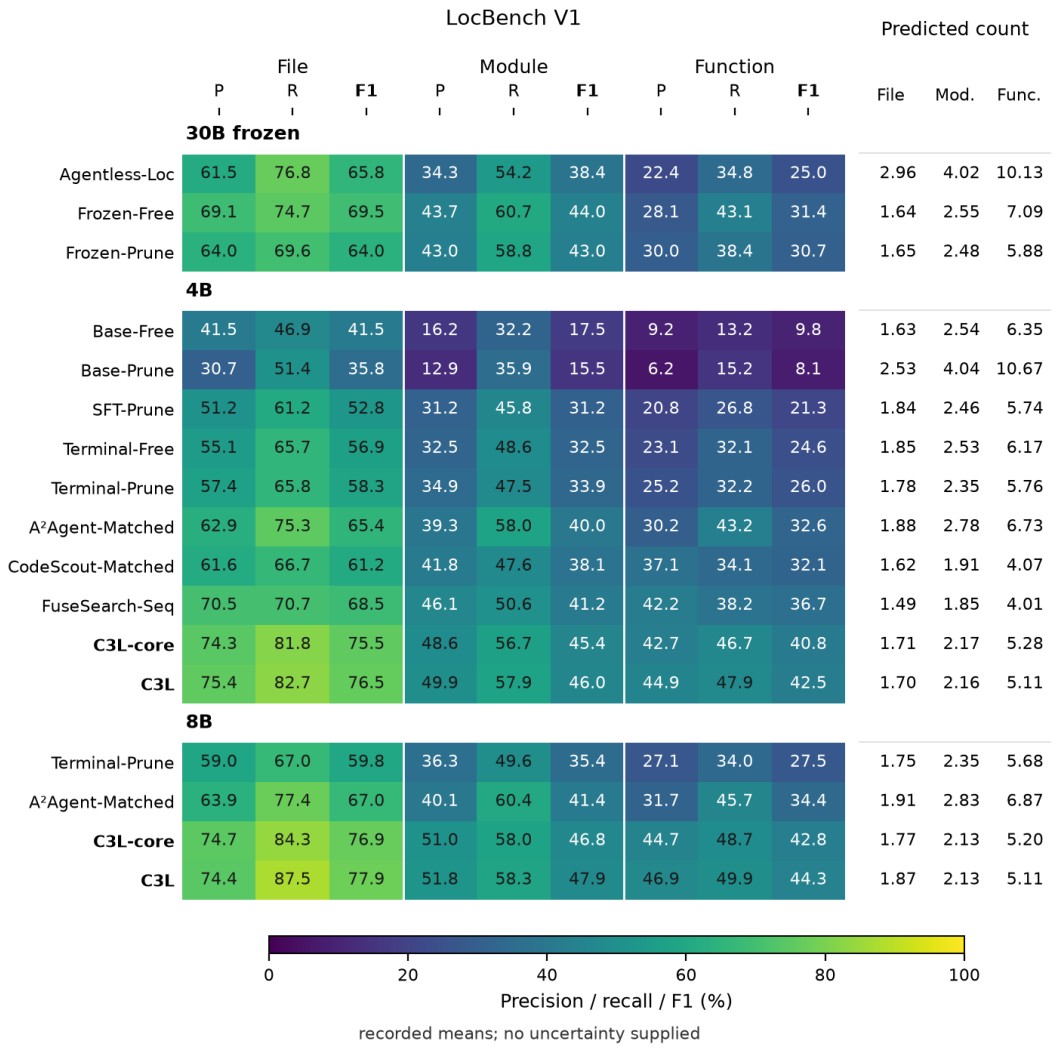


Figure A.18: Three-granularity localization on LocBench V1. Macro precision, recall and F1 and predicted set sizes for the same 17 configurations, formatted as in Figure A.16.

A.8 Trajectory Case Studies

From online trajectories we select five kinds of decision and follow each from the evidence, through the declaration and the immediate change in coverage, to the final outcome. Every case satisfies the selection condition that the turn discovers no new reference entity. The cases show where the credit for an exclusion or a keep comes from. Each row is one event on one trajectory, and the table is not an estimate of how often the event occurs.

In the correct-exclusion case, the agent reads call sites and the decorator chain, then deletes a pair of synchronous adapter implementations and their helper methods. Volume falls by 3.14 points and the reference ceiling is unchanged (Table A.20). The turn hits no new reference entity, and it removes candidates that later turns would otherwise have to distinguish, so it receives the volume reward. The keep action, at turn 4, restricts files to two module directories. It spends 0.0071 of reference ceiling for a 71.35-point drop in volume, and the final prediction still matches the files and the functions completely. The two cases show the different scales at which item-wise exclusion and set-level keep act during search.

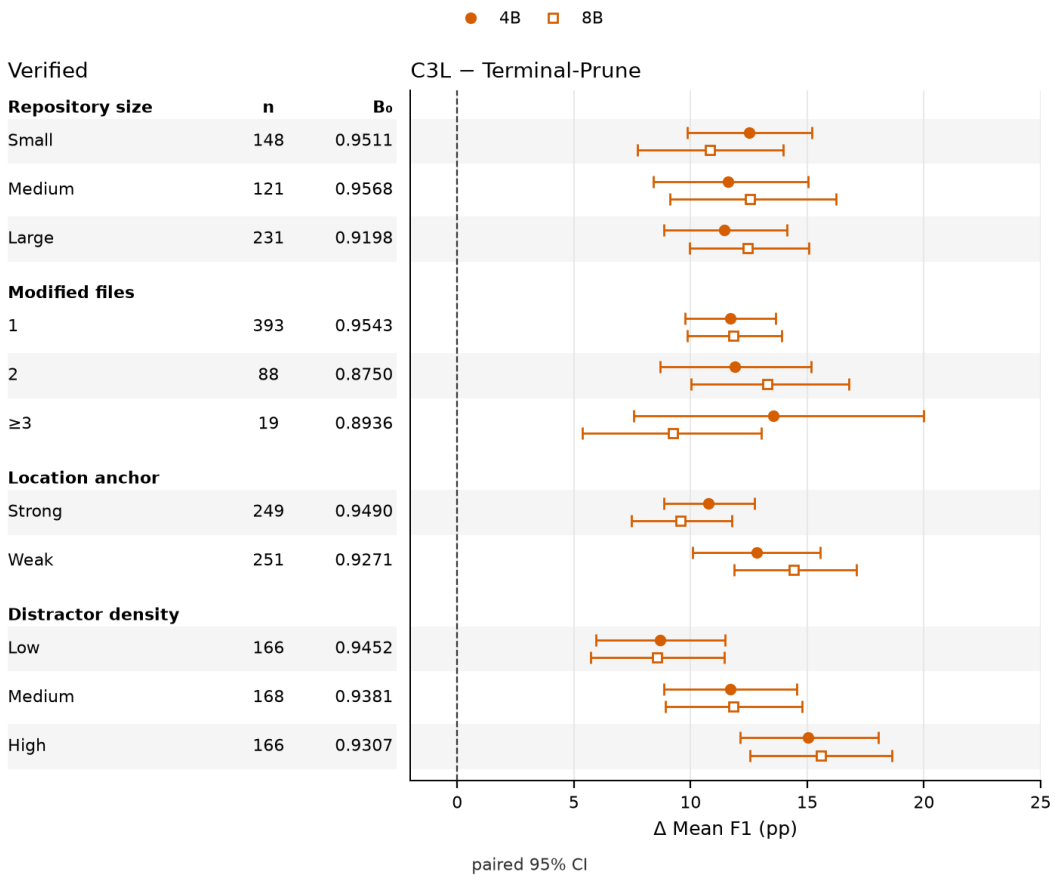


Figure A.19: Localization gains across Verified strata. C3L minus Terminal-Prune Mean F1 with issue-paired 95% CIs using 3 seeds. Circles/squares denote 4B/8B. Each stratum lists issue count and initial reference ceiling. Different stratification axes may contain the same issues and are not independent replications.

The mistaken exclusion and the partial-coverage case show why the suffix should decay with the coverage that remains. In Table A.21, the trajectory that excludes the only reference file has $d = 0$ on later turns, and no later action can raise its state F1 above 0. The trajectory that deletes one of three reference files keeps $d = 0.8000$, and later state F1 rises by 18.75 points. The action that caused the loss is weighted from the state before that action. Later actions that can still improve the remaining targets receive a continuous reduction in weight. The distinction matches the result in Appendix A.5: continuous decay outperforms a hard cutoff.

The last case excludes the subclass file edited by the reference patch and keeps the shared implementation in the base class. Under the original reference label the terminal score is 0 and later $d = 0$. The frozen repairer then generates a patch inside the retained scope. After that patch passes F2P/P2P, the actual edit locations enter the cache, the score rises to 0.8889, and the recomputed file ceiling and d return to 1. The event joins the candidate decision, the execution check, and the within-batch recomputation of credit. Scoring a search action has to count the valid repair path that the action kept.

A.9 Failure Analysis

C3L scores below FuseSearch-Seq on 161 of the 500 Verified issues (Table B.6). Table A.22 assigns each loss to one cause. The first two causes mark issues that neither method can fully cover; the

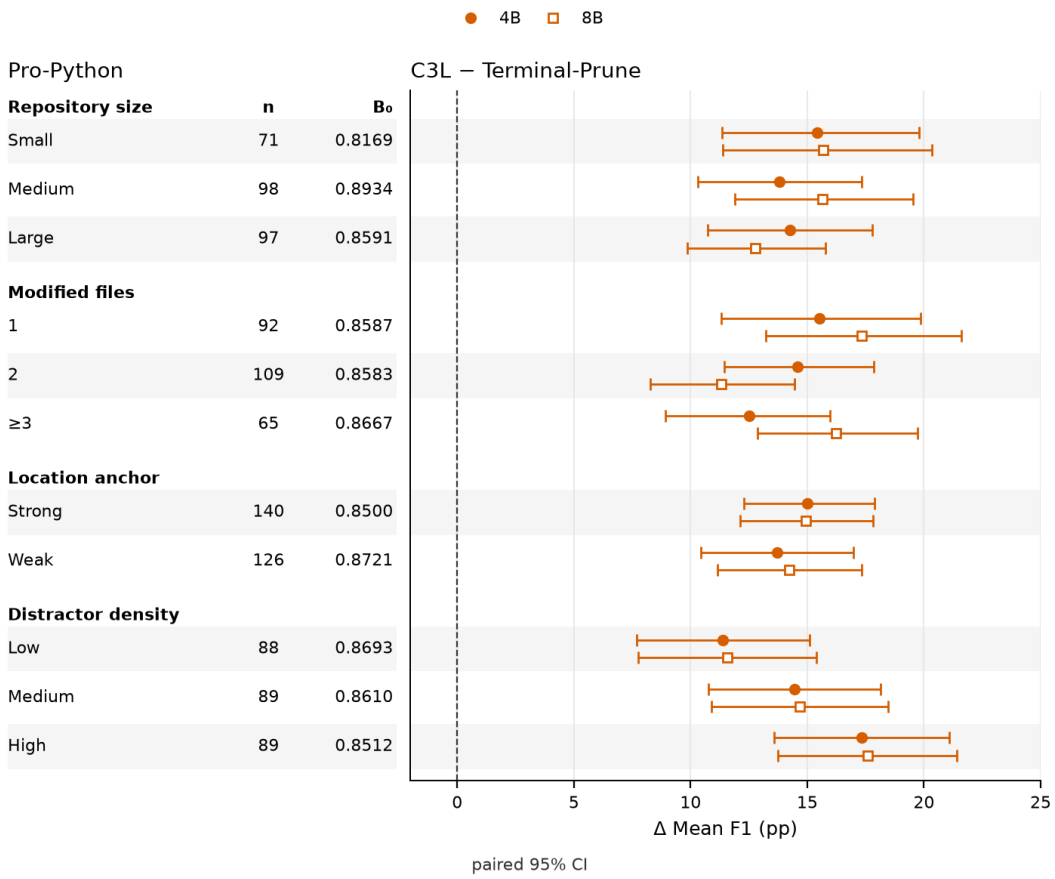


Figure A.20: Localization gains across Pro-Python strata. C3L minus Terminal-Prune Mean F1 with issue-paired 95% CIs using 3 seeds. Circles/squares denote 4B/8B; counts and initial reference ceilings are shown per stratum. Different axes may overlap in issues and are not independent replications.

next three describe C3L's own decisions: excluding a reference file, over-pruning finer granularities after keeping the right files, and keeping too many survivors.

Fifty losses (31%) fall on issues that neither method can fully cover: 46 require a new entity and 4 are retrieval misses. Both methods face the same ceiling there, and the per-issue gaps (-14.2 and -11.6 points) are smaller than the average loss of -17.9. The largest decision-level cause is excluding a reference file (38 losses), which is also the costliest at -28.9 points per issue, because a lost file removes its modules and functions with it. This is the error that coverage loss and suffix decay target, and the remaining PER of 11.49% (Table 3) shows where further gains lie. Over-contraction at finer granularities (34 losses) and excess survivors (21) cost less per issue, and the 18 residual losses have no single dominant cause.

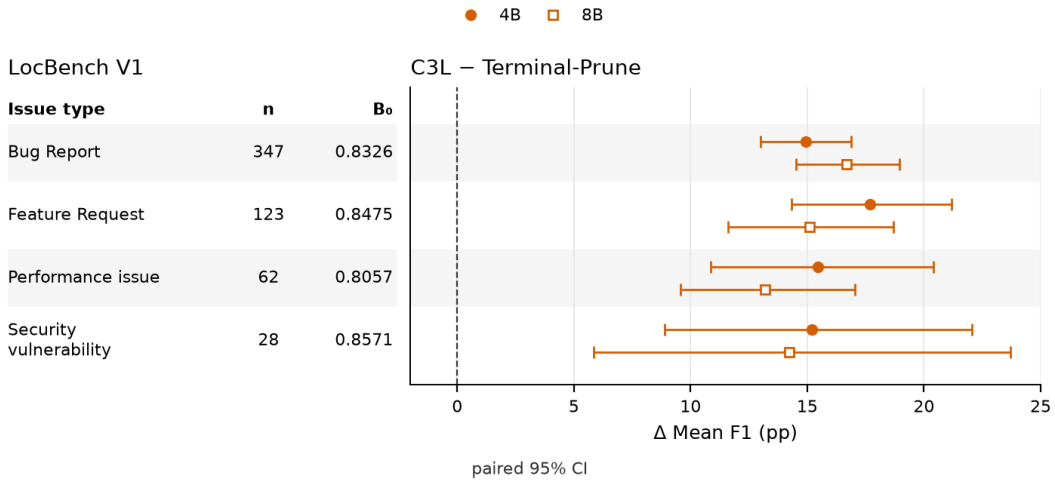


Figure A.21: Localization gains across LocBench V1 strata. C3L minus Terminal-Prune Mean F1 with issue-paired 95% CIs using 3 seeds. Circles/squares denote 4B/8B. Counts and initial reference ceilings accompany each stratum, including official issue types. Different axes may overlap in issues and are not independent replications.

Table A.20: Observed contraction actions and their immediate effects.

Case	Turn	Evidence	Decision	ΔV (pp)	ΔB ref	Ref. found
Correct distractor exclusion	7	Code reading: call sites + decorator chain	Exclude: 2 synchronous adapters + 3 helper methods	-3.14	0.0000	No
One-step scope restriction	4	Keyword search + symbol lookup	Keep: files in 2 module directories	-71.35	-0.0071	No
Reference-target exclusion	9	Directory browsing: name-based inference	Exclude: serialization module with only reference file	-1.87	-1.0000	No
Progress after partial coverage loss	6	Code reading: exception propagation	Exclude: 1 of 3 reference files + all its functions	-2.45	-0.2000	No
Validated alternative repair	11	Code reading: base class + subclass overrides	Exclude: subclass file modified by reference patch	-1.32	-1.0000	No

Note. Five selected events from online trajectories. Turn numbers refer to positions within the 20-call budget. ΔV is in percentage points of initial volume; ΔB is the dimensionless change in the reference ceiling. Absence of a new reference discovery is a selection condition, not a frequency estimate.

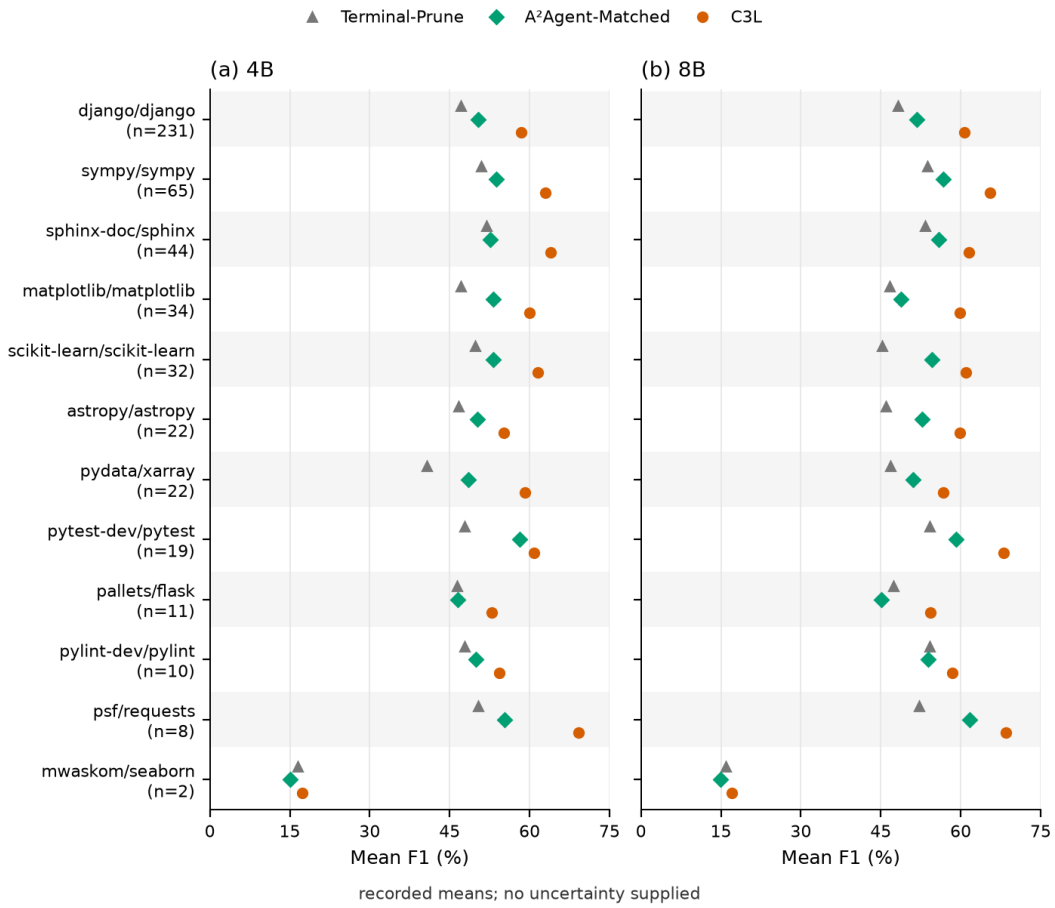
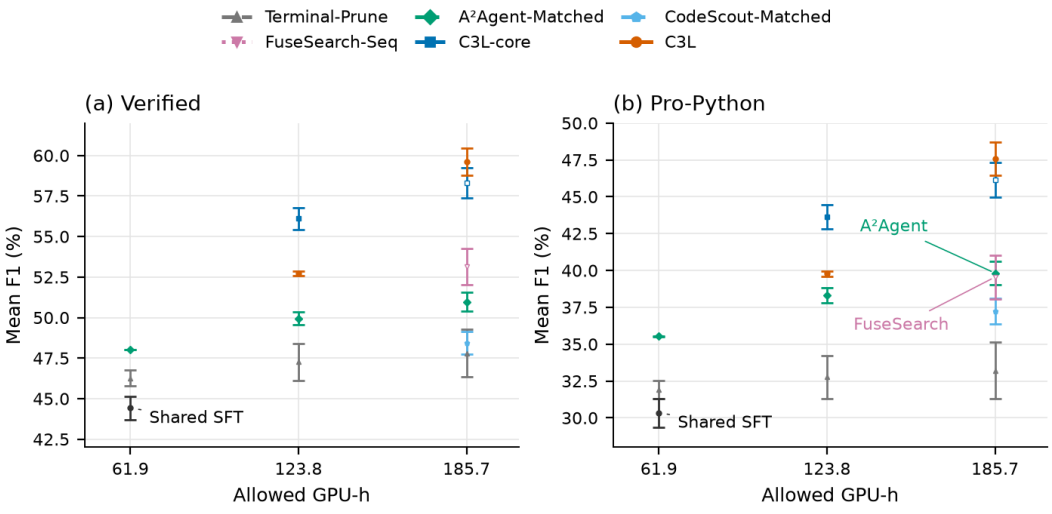


Figure A.22: Localization in each Verified repository. Means across 3 seeds for 4B and 8B locators in 12 repositories. Labels include issue counts, including the two-issue seaborn subset. Repository-balanced aggregates appear in Table A.17.

Table A.21: Subsequent credit and outcomes of the selected trajectories.

Case	Mean d	ΔF1 (pp)	R before	R after	F2P/P2P	Final outcome
Correct distractor exclusion	1.0000	—	1.0000	1.0000	Not triggered	All 3 granularities matched; 2 files / 4 functions
One-step scope restriction	1.0000	—	0.8333	0.8333	Not triggered	File/function matches complete; 1 peripheral module label missed
Reference-target exclusion	0.0000	0.00	0.0000	0.0000	Failed	All reference files lost; later weights zero
Progress after partial coverage loss	0.8000	+18.75	0.8000	0.8000	Failed	2/3 reference files retained; functions within them matched
Validated alternative repair	0.0000	0.00	0.0000	0.8889	Passed	Shared base-class implementation localized and repaired

Note. Cases follow Table A.20 in the same order. Mean d is the mean subsequent decay factor and uses the original reference labels: in the alternative-repair case it is 0.0000 before calibration and becomes 1.0000 after accepting the validated label. ΔF1 is the post-loss state change under the reference label, excluding the loss turn, and is undefined (—) for the first two cases; once a loss drives the reference ceiling to 0, state F1 stays at 0, so ΔF1 is exactly 0.00. R before and R after are the terminal scores before and after calibration, on a 0–1 scale. These are individual cases without group-level uncertainty.



Actual resources at each allowed budget

	Allowed: 61.9			Allowed: 123.8			Allowed: 185.7		
	GPU-h	CPU-h	Iters	GPU-h	CPU-h	Iters	GPU-h	CPU-h	Iters
Terminal-Prune	58.9	16.0	1	113.7	32.0	2	168.4	48.0	3
A²Agent-Matched	60.7	16.0	1	117.2	32.0	2	173.7	48.0	3
CodeScout-Matched	—	—	—	—	—	—	178.5	48.0	3
FuseSearch-Seq	—	—	—	—	—	—	162.9	48.0	3
C3L-core	Shared SFT			122.2	32.0	2	181.2	48.0	3
C3L	4.2	0.0	0	64.7	95.6	1	185.7	244.0	3

Figure A.23: Quality at fixed cumulative GPU-hour budgets. $\text{mean}_{\pm\text{SD}}$ across 2 seeds, using only completed checkpoints. H=185.7 GPU-h is the 3-seed mean cost of full 4B C3L. Core/full share the SFT point at H/3 (4.2 GPU-h, 0 CPU-h, zero completed post-SFT iterations). Em dashes mark budgets without a completed checkpoint. Verified C3L-minus-Terminal-Prune paired differences are -1.85 pp at H/3 (95% CI [-2.81, -0.88]) and +11.81 pp at H ([10.05, 13.60]); these CIs are distinct from absolute-score SD. GPU and CPU costs are reported separately.

Table A.22: Attribution of the 161 Verified issues on which 4B C3L scores below FuseSearch-Seq.

Cause	**Condition on C3L's terminal state**	**Issues**	**Mean ΔF1 (pp)**
New entity required	Reference contains an entity absent from the pre-fix repository	46	−14.2
Retrieval miss	Reference representable but not fully covered by C_x	4	−11.6
Reference file excluded	$y_f \subseteq C_x^f$ but $y_f \not\subseteq S_T^f$	38	−28.9
Over-contraction	All reference files kept; lower module or function recall than FuseSearch-Seq	34	−16.5
Excess survivors	All reference files kept; lower precision than FuseSearch-Seq	21	−12.3
Empty prediction	$S_T = \varnothing$	0	—
Residual	None of the above	18	−14.6
Total		161	−17.9

Note. Causes are assigned per issue in the listed order; an issue enters a cause when at least two of three seeds meet its condition. Mean ΔF1 is C3L minus FuseSearch-Seq Mean F1 within the cause. At most 114 losses can require a new entity and at most 12 can be retrieval misses (Appendix A.1); C3L returns no empty prediction on Verified.

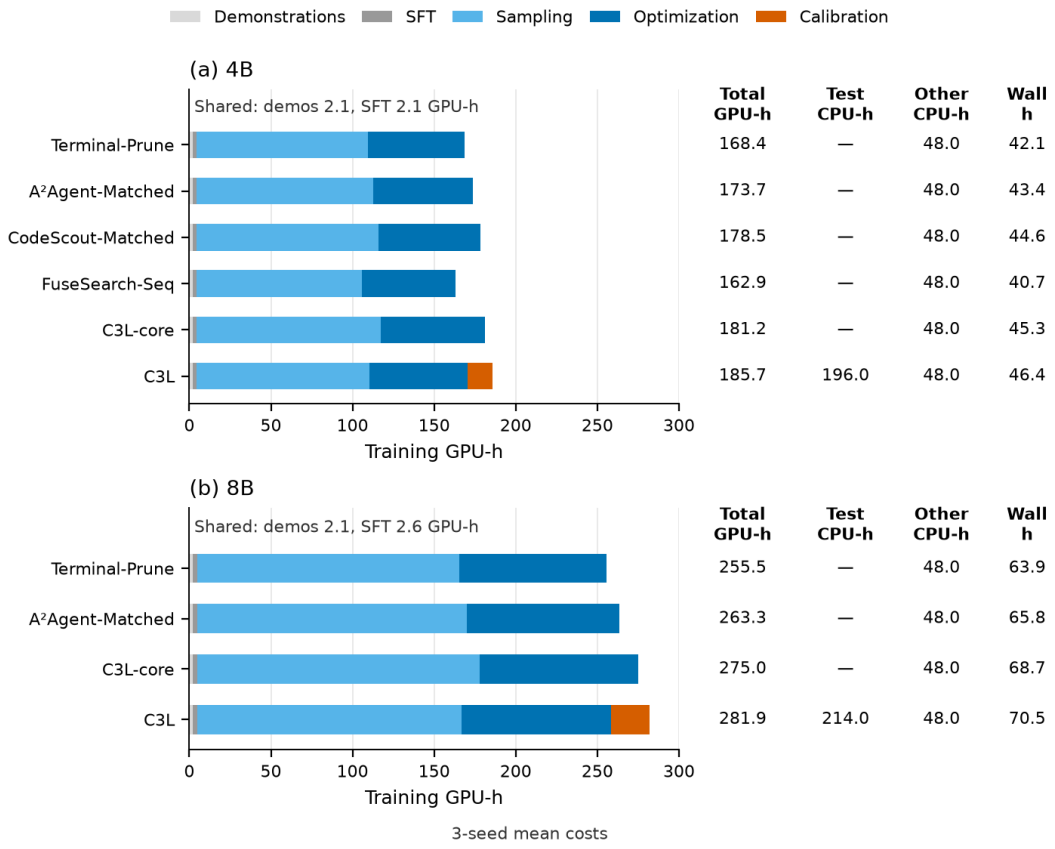


Figure A.24: Training cost by computational stage. Mean cost across 3 seeds. GPU-hours are decomposed into five stages; shared demonstration/SFT costs are shown per model block. Total GPU-hours include all stages. Test CPU-hours, other CPU-hours and wall-clock hours are separate quantities; em dashes denote not applicable values.

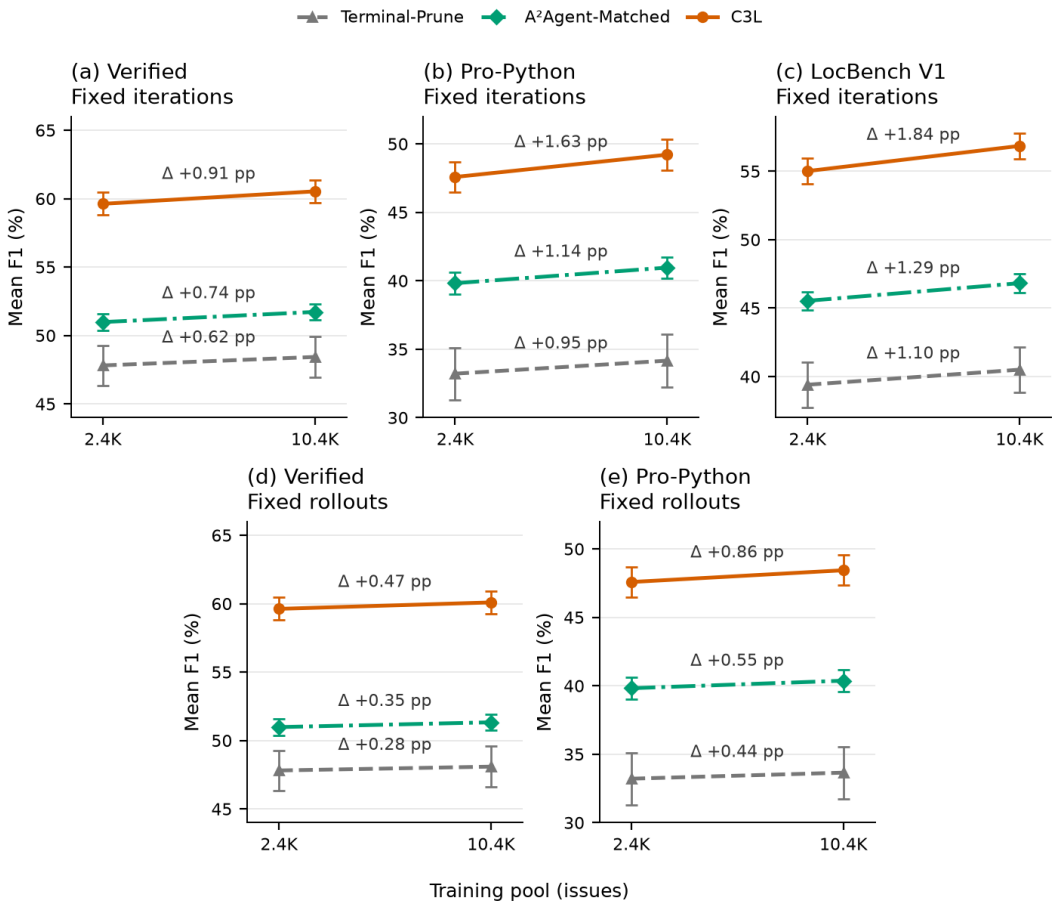


Figure A.25: Training-pool expansion under two rollout protocols. $\text{mean}_{\pm\text{SD}}$ across 2 seeds. The upper row uses three full iterations (57,600/249,600 rollouts for 2.4K/10.4K pools); the lower row fixes 57,600 rollouts per seed and visits 2,400/5,666 distinct issues. The 2.4K baseline is shared across protocols. LocBench has no fixed-rollout result. Delta labels are differences of means; fixed rollouts do not imply equal GPU cost.

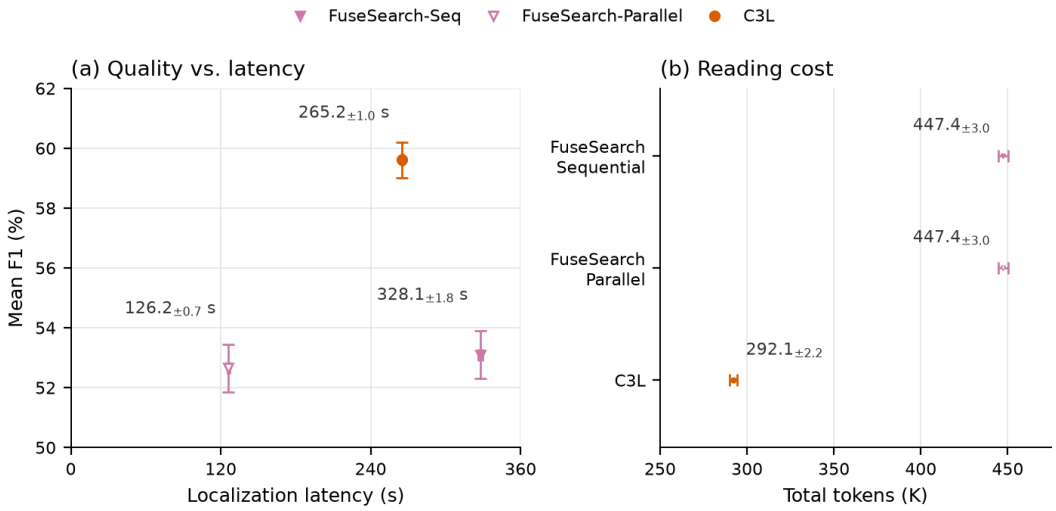


Figure A.26: Sequential contraction and parallel tool execution. Verified with an allowed total budget of 20 tool calls; $\text{mean}_{\pm\text{SD}}$ across 3 seeds. Latency/F1 axes show marginal SD; token cost retains its SD. FuseSearch-Seq/Parallel/C3L use concurrency 1/4/1, actual calls 18.13/18.31/13.35, interaction turns 18.13/5.79/13.35 and training costs 162.9/167.3/185.7 GPU-h. Parallelism and cumulative reading cost are evaluated separately.

B Experimental Details

B.1 Method Details

Protocol. Entities use canonical identifiers that include their granularity, and the environment expands declarations deterministically onto the current survivors. Removing a file removes its modules and functions, removing a module removes its functions, and removing a function leaves its parents in place. A keep declaration contracts an entire granularity once the evidence has narrowed it to a few locations. Every surviving candidate enters the answer, and every explored location that does not survive corresponds to an explicit, attributable contraction. Candidate decisions and tool calls share one output format that the environment parses and executes; because an untrained instruct model does not follow this protocol reliably, the shared SFT stage precedes training.

Volume and ceiling. The weights q_x normalize fused retrieval scores within each nonempty granularity, mix granularities with equal weight, retain a small uniform mass, and stay fixed as a rollout contracts (Appendix B.3). The logarithm in Φ_x^{vol} pays the same for the same proportional cut (halving the volume from 1 to $1/2$ pays as much as halving it from $1/10$ to $1/20$); $\varepsilon_v > 0$ keeps the potential finite at $V_x(S) = 0$ and caps the contraction reward of a trajectory at $\log(1 + 1/\varepsilon_v)$. The per-granularity ceiling is

$$b_g(S; y') = \begin{cases} 2\rho_g(S; y')/\big(1 + \rho_g(S; y')\big), & |y'_g| > 0, \\ 1, & |y'_g| = 0. \end{cases}$$

Because b_g is computed per granularity, B_x ignores hierarchical coupling (emptying a module also empties its functions) and upper-bounds reachable F1; the two properties used in the paper, monotonicity along a trajectory and $R_x(S_T) \le B_x(S_T)$, hold regardless. Since no subset of S_T scores higher than $S_T \cap y'$, $R_x(S_T) \le B_x(S_T)$, and the two process signals divide the work: coverage loss protects the ceiling $B_x(S_T)$, and contraction reward closes the precision gap $B_x(S_T) - R_x(S_T)$ left by candidates that were not cut.

Reward scale and κ_x. The coefficient κ_x takes the higher value when the issue text, a traceback, or an F2P test points directly at the reference module, where excluding that module is almost certainly wrong, and the lower value otherwise, where the reference patch is more likely one of several viable repairs; like $\mathcal{Y}_x$ and d, it uses privileged information only during training (Appendix B.5). The three reward terms share a common scale: $R_x \in [0, 1]$, the contraction rewards of a trajectory sum to at most $\alpha \log(1 + 1/\varepsilon_v)$, and a complete loss of coverage costs $\kappa_x B_x(S_0)$. We choose $\alpha \log(1 + 1/\varepsilon_v) < \kappa_{\mathrm{low}}$, so on every issue whose initial candidates cover the reference, no amount of contraction can buy back a lost repair path, while the contraction incentive stays on the scale of the terminal score. On a single cut, κ_x sets the exchange rate between the volume removed and the coverage given up; Appendix A.5 scans this rate.

Grouping and advantages. The directory summary alone rarely changes, because clearing an entire directory is rare, so the volume bin separates decisions that share a directory distribution but differ in how far contraction has gone; when $S = \varnothing$, ν takes the issue's maximum bin index. Budget bins keep turns near termination apart from turns that still have room to explore. The turn advantage is $\hat{A}_{i,t} = (G_{i,t} - \mathrm{mean}_{\mathcal{H}_{i,t}} G)/(\mathrm{std}_{\mathcal{H}_{i,t}} G + \varepsilon_a)$; if a group contains fewer than two independent rollouts or its return SD is at most ε_g, only $\hat{A}_{i,t}$ is set to zero. Both advantages use population SDs, and $\omega_0, \omega_1 \ge 0$. The signature encodes directory distribution and remaining volume but not coverage, so one group can contain rollouts that have lost a reference location and rollouts that have not: $\hat{A}$ ranks turns within the group, while d applies an absolute reduction based on each trajectory's own remaining coverage.

Suffix decay and objective. The file-level ceiling is $B_x^f(S) = \max_{y' \in \mathcal{Y}_x} b_f(S; y')$. Without normalization by $B_x^f(S_{i,0})$, a trajectory whose candidates miss a file from the start would be down-weighted from its first turn. Because $d_{i,t}$ is computed before the action, the exclusion that lowers coverage is priced through its own reward, and while some coverage remains, later decisions can still improve the answer and are never fully masked. The clip range also bounds the advantage: values outside $[\tau \log w_{\min}, \tau \log w_{\max}]$ no longer change the weight, an interval calibrated in within-group

Table B.1: Models and their experimental roles.

Model	Status	Role
Qwen3-4B-Instruct-2507 (Qwen Team, 2025a)	Trainable	Main locator for experiments and ablations
Qwen3-8B (Qwen Team, 2025b)	Trainable	Scale comparison for localization, repair and inference cost
Qwen3-Coder-30B-A3B-Instruct (Qwen Team, 2025c)	Frozen	SFT demonstrations, reference locator and main downstream repairer
Qwen2.5-Coder-32B-Instruct (Qwen Team, 2024)	Frozen	Training-time label calibration and second downstream repairer

Note. All stages use non-thinking generation. Model identifiers link to the corresponding model releases.

Table B.2: Datasets and evaluation scopes.

Dataset	Use
SWE-smith (Yang et al., 2025)	Train 2,400 + validation 267; expanded train pool 10,400
SWE-bench Verified (Chowdhury et al., 2024)	All 500: localization, repair and analysis
SWE-bench Pro (Deng et al., 2025)	266 Python issues from 731: complex-repository generalization
LocBench V1 (Chen et al., 2025)	All 560: generalization across issue types

Note. Training and test data are separated by repository; training and validation are separated by issue. Evaluation retains candidate misses and newly introduced entities.

SD units, and $w_{\max}/w_{\min}$ sets the learning-strength gap between the best and worst decisions. The loss masks system prompts, the issue, tool returns, and padding; dividing by $|\mathcal{V}_i|$ gives every trajectory equal influence regardless of length, and N is the number of trajectories collected in the iteration.

Label cache. A passing patch yields a canonical location set y^{alt} and the update $\mathcal{Y}_x \leftarrow \mathcal{Y}_x \cup \{y^{\text{alt}}\}$. Because $R_x = F(S_T, y^{\text{alt}})$ still scores the terminal set, the cache never rewards leaving a large set for the repairer to salvage, and a failed validation leaves the trajectory under the current labels. Scopes are validated in decreasing order of $B_x(S_T)$; the validation cap bounds patch generation and test execution, and the label cap keeps the signal discriminative, since R_x and B_x both maximize over $\mathcal{Y}_x$. The acceptance test (scope, F2P, and P2P) is fixed for each instance and does not change with the policy, so the cache grows only where a viable repair exists.

B.2 Models and data

Every stage sets `enable_thinking=false`. Localization inference runs only the locator. Demonstration generation, training-time calibration, and downstream repair are accounted separately. The calibration repairer is a different model from the repair backend in the main text. Table B.1 lists the role of each model. Table B.2 lists the datasets and how each is used.

Split and deduplication. After SWE-Smith mirror names are mapped back to canonical `owner/repo`, we start from 50,908 Python instances in 131 repositories and remove the 14 repositories that overlap any test set, together with their 9,232 instances. This leaves 41,676 instances in 117 repositories. After deduplicating issues and reference patches and filtering for trainable labels, a fixed split seed draws 2,667 instances stratified by repository, which are then split into training and validation by issue. The scaling experiment adds 8,000 further distinct issues from the same eligible

pool. The two training pools cover 112 and 117 repositories. The validation set stays fixed across models, seeds, and the scaling setting.

Labels and representability. Location labels are extracted from the reference repair patch. `test_patch` is not used as a location label. A module follows the dataset's class-level and module-level symbol semantics. Each entity is mapped to an identifier containing its granularity, canonical path, and qualified symbol name, and the necessary parents are filled in. Directories and packages are used only for grouping. An entity that already exists, or that the patch deletes, uses its pre-fix location. A newly added entity stays in the scoring label and is marked for representability. A file-only edit does not invent a module or a function. A missing record in a published label is not read as a true empty label. Main results use the full test sets. The representable subset and the initially fully covered subset in Appendix A.1 each keep their own denominator.

B.3 Candidate construction and interaction protocol

Retrieval and weights. Candidates are built from the issue and the pre-fix repository alone. BM25, symbol-name matching, and path matching each return a list. Their union is ranked by reciprocal rank fusion with constant 60. A missing rank contributes 0, and ties break by canonical identifier. We first keep at most 300 files, then at most 2,000 functions inside those files. Module candidates include the modules and classes in the selected files and the parents of the selected functions. A small repository keeps however many entities it actually has. For each granularity g whose candidate set is nonempty, the fused scores are normalized to $\bar{s}_g(a)$. If every score in that granularity is zero, the distribution is uniform. Let $\mathcal{G}_x^+$ be the set of granularities with nonempty candidates. The frozen weight is

$$q_x(a) = \frac{1}{|\mathcal{G}_x^+|}\left(0.95\bar{s}_g(a) + \frac{0.05}{|C_x^g|}\right), \qquad a \in C_x^g.$$

Each nonempty granularity retains 5% uniform mass, so every candidate weight is positive, and the weights are not updated as the set contracts. When the candidate universe is empty, the environment emits the empty set, counts the issue in the main evaluation, reports it as its own diagnostic, and does not compute an undefined volume reward.

Candidate catalog. The catalog is hierarchical so that it fits the context budget. The initial prompt lists only the candidate files, as a directory tree with their canonical identifiers and the number of module and function candidates in each file, which takes a few thousand tokens at the default cap. The symbol-lookup tool returns the canonical identifiers of a file's surviving module and function candidates on demand, and code is read on demand. Because declarations also accept path patterns, the agent can exclude or keep whole files and directories without expanding them. After each declaration, the environment appends a compact survivor summary (per-granularity counts and the surviving files, with a directory collapsed to one line once it holds more than 20 surviving files), and only the latest summary is kept in the context.

Actions and hierarchical coupling. Each turn calls one read-only tool and, after the observation, declares an exclusion set X_t and an optional keep set K_t. If K_t is omitted, it is the survivor set from the previous turn. The environment expands canonical identifiers and path patterns deterministically. Deleting a file or a module also removes its candidate children. Deleting a function leaves the parents. An empty declaration and a repeated declaration leave the state unchanged. A removed entity can still be read, and it cannot rejoin the survivor set. Tool returns are truncated in a stable order, at most 2,048 tokens at a time. The catalog, the survivor summaries, and the decision declarations all count as interaction cost.

Stopping rules. The trajectory stops when it hits the tool-call budget, the output budget, or the context budget. Prune serializes S_T after the last complete legal decision and does not emit a separate answer list. Free takes the last complete submission, or the empty set if none was submitted. SFT demonstrations for the two protocols use the same declaration syntax. The independent subset-output variant in Appendix A.5 intersects the last complete declaration with S_T, and that declaration spends budget as well. The `restore` variant treats C_x as the upper limit on what can return, and expands the action under the same hierarchy.

Table B.3: Shared training and inference configuration.

Configuration	Default
Parameter updates	LoRA $r = 16,\ \alpha = 16$; attention + MLP linear layers
SFT	3 epochs; learning rate 2×10^{-4}
Post-SFT training	3 iterations; 2 epochs/iteration; 8 rollouts/issue; C3L samples each iteration from one frozen snapshot, GRPO/GSPO methods refresh it every 200 issues (12 rounds)
Optimizer and schedule	AdamW; learning rate 5×10^{-5}, weight decay 0.01; 10 warmup steps/stage, then linear decay
Effective batch	16 trajectories; token-normalized loss on agent tokens; other tokens masked
Decoding	Train temperature 1.0, top-p 0.8, repetition penalty 1.05; greedy evaluation
Budget	≤ 20 tool calls; $\leq 8{,}192$ output tokens; 32,768 context tokens/call
Numerical precision	BF16; FP32 rewards, advantages and weights
Seeds and checkpoints	3 training seeds + 1 split seed; checkpoint each iteration; select validation Mean F1 (earlier on tie)
Environment	4× RTX Pro 6000 (96 GB/card); 128 logical cores; 512 GB RAM; Transformers/PEFT/DeepSpeed + vLLM; retain versions and peak GPU memory

Note. This is the main configuration. Two-seed design scans and scaling experiments are identified in their own captions; frozen audits use one seed. Checkpoint selection uses the fixed validation split.

Table B.4: Default C3L parameters and comparison rules.

Method parameter	Default / rule
Contraction weight α	0.1; bound $\alpha \log(1 + 1/\varepsilon_v) \approx 0.69$, comparable to $R_x \in [0, 1]$
Coverage-loss weights	$\kappa_{\text{low}} = 1;\ \kappa_{\text{high}} = 2$
Stabilizers	$\varepsilon_v = 10^{-3};\ \varepsilon_a = \varepsilon_g = 10^{-6}$
Advantage fusion	$\omega_0 = \omega_1 = 0.5$; undiscounted returns
Weight temperature and range	$\tau = 1;\ [w_{\min}, w_{\max}] = [0.2, 5]$; equivalent to ± 1.61 within-group SD clipping; no extra A clipping
Suffix decay	$d_{i,t} = B_x^f(S_{i,t-1})/B_x^f(S_{i,0})$, pre-action; use 1 and report separately if $B_x^f(S_{i,0}) = 0$
State signature	$\sigma(S) = (\psi_1(S), \nu(S))$: top-level directories + volume bin; root files assigned to ROOT
Remaining-budget bins	Calls remaining before action: 1–5, 6–10, 11–15, 16–20
Valid comparison groups	Same issue/snapshot; ≥ 2 independent rollouts; return SD $> \varepsilon_g$; zero turn term for degenerate groups
Validation budget	$\leq k = 2$ terminal scopes/issue/iteration, ranked by $B_x(S_T)$; $\lvert\mathcal{Y}_x\rvert \leq 4$

Note. Advantages use population SD; experimental error bars use sample SD across seeds. States and suffix factors are evaluated before each action. Rewards, advantages and learning weights are stop-gradient.

B.4 Training configuration and method parameters

Shared SFT. On each training issue the frozen teacher runs Free and Prune separately and samples 4 trajectories of each. We keep a trajectory that is legal, has the highest reference F1, and hits at least one reference entity. A tie takes the earlier sample index. An issue enters the mixture only when both protocols have a valid demonstration, one trajectory each, so the mix is 1:1. The teacher does not read the reference patch or the hidden tests. Labels are used only to select demonstrations offline. Prune's contraction actions are generated online. Each backbone and seed receives one shared checkpoint from this procedure. The demonstrations do not depend on the backbone or the seed, so the teacher generates them once and each of the six runs is charged an equal share of their cost (2.1 of 12.6 GPU-h; Figure A.24). A protocol prompt distinguishes the task. Candidates and tools match. Later training uses the full trainable set, not only the issues on which a demonstration succeeded.

The shared training configuration and the default C3L parameters are listed in Table B.3 and Table B.4.

C3L uses the imitation-weighted objective of § 3.3. It adds neither importance-ratio clipping nor a reference-policy KL. Each iteration freezes the policy and samples again. The main setting does not reuse trajectories from earlier iterations. Grouping and suffix decay use the state before the action. Rewards, advantages, and weights are stop-gradient. Advantages are standardized with the population standard deviation. Several turns of one rollout can fall in the same group, and those turns cannot by themselves meet the requirement of at least two independent rollouts. A degenerate group zeros only the turn advantage and keeps the trajectory component. The default signature targets similar residual tasks. It does not require the evidence already read to match.

The main configuration was fixed on the validation split before any test run, by one-factor sweeps of α, the κ multiplier, and ω_1 (seven single-seed runs); τ and the weight range were set a priori, and C3L uses the central learning rate. Once fixed, it is used for every official seed. The scans in Appendix A.5 are not fed back to choose a configuration from test scores. A baseline that keeps its native objective uses the default parameters of the locked release. Every trained baseline receives the same three-point validation budget, at $\{0.5, 1, 2\}$ times the central learning rate. The main training results use 3 seeds. Scans of the reward, the grouping, and the decay, together with the pool-expansion and equal-compute quality comparisons, use 2 seeds; the history-key grouping is rerun with a third seed for its paired comparison with C3L-core. Protocol constraints use 3 seeds. Software versions and peak GPU memory are stored with the run record.

B.5 Label calibration and downstream repair

Anchors and penalty strata. An issue is marked a strong anchor, and receives κ_{high}, when an explicit path or a qualified symbol in the issue text or the traceback maps directly onto the reference module, or when a direct module import in an F2P test points at the reference module. Every other issue receives κ_{low}. The rule sets the training reward only. It does not show the localization agent the reference module, the F2P tests, or the anchor label. Strata used at evaluation are computed offline after the rollout ends. The uniform-coefficient control uses the mean of the training-set coefficients, $\bar{\kappa} = |\mathcal{D}_{\text{train}}|^{-1} \sum_x \kappa_x$.

Cache trigger and validation. Each training configuration and each seed keeps its own $\mathcal{Y}_x$, initially the reference label alone. A terminal state is judged against the label set at the start of the batch. Trajectories with $B_x^f(S_T) < 1$ enter the validation queue. Inside one issue and one iteration, terminal scopes are deduplicated, then at most $k = 2$ independent requests are issued, in decreasing order of $B_x(S_T)$. Triggered trajectories that fall outside the budget stay in the audit denominator. Expansion stops once four acceptable labels have been stored. An issue whose initial candidates are incomplete can also trigger, and those issues are reported separately as retrieval misses. Trajectories that share a scope share the request result. Trajectory counts, call counts, and the number of distinct canonical label sets are recorded separately.

Scope check and recomputation. The edited file must survive. An edit inside an existing module or function also requires that module or function to survive. Survival of the file does not bypass a deleted function. A gap at file or module level can receive a new entity. The frozen calibration repairer sees only the issue, the allowed locations, and the pre-fix code. The actual patch must pass the scope check, F2P (fail-to-pass: failing before the fix and passing after it), and P2P (pass-to-pass: passing both before and after the fix). On a pass, the alternative label is taken only from the actual diff. New entities are not inserted into the localization candidates. After the labels that passed in the batch have been collected, the updated $\mathcal{Y}_x$ is frozen, and R_x, B_x, r_t, and $d_{i,t}$ are recomputed together for every trajectory of that issue before advantages are formed. The cache does not enter the localization input at test time, and it does not enter the official score against the reference label.

Fixed downstream procedure. Every locator uses the same Agentless backend and the same context packer. Locations and code are packed in a stable order of canonical identifiers, truncated by the same rule, and the truncation rate is recorded. Each issue produces one greedy patch, with at most 8,192 output tokens and 32,768 context tokens. The repairer does not localize again, and it does not regenerate from test feedback. The edit-scope check uses the hierarchy above. Tests and timeouts run in the locked benchmark harness. A reference-location input and training-time calibration use this same repair procedure and the same generation budget.

Table B.5: Baseline protocols, training objectives and comparison roles.

Method	Output protocol	Training signal and objective	Comparison role
Retrieval	Fixed ranking	No training; top-5 fused entities/granularity + parents	Low-cost retrieval reference
Agentless-Loc	Fixed two-stage pipeline	No training; frozen Qwen3-Coder-30B-A3B; Agentless file ranking + per-file extraction	Standard untrained pipeline
Base-Free / Base-Prune	Free / Prune	Instruct model; no project-specific training; protocol prompts differ	Base-model capability and protocol adaptation
Frozen-30B-Free / Frozen-30B-Prune	Free / Prune	Frozen Qwen3-Coder-30B-A3B; no training	Larger frozen-model reference
SFT-Free / SFT-Prune	Free / Prune	Shared mixed-protocol SFT checkpoint	Protocol adaptation and initialization
Terminal-Free / Terminal-Prune	Free / Prune	Terminal F1 + trajectory advantage; imitation-weighted, $d = 1$; no process terms, turn advantage, decay or cache	Same-objective terminal control; isolates contraction protocol
A^2Agent-Matched	Independent submission	First discovery + Discovery-Commit + terminal F1; history $H = 2$	Discovery-based turn credit
CodeScout-Matched	Independent submission	Terminal three-granularity F1; original GSPO	Trajectory-level RL
FuseSearch-Seq	Independent submission; sequential tools	Coupled quality/novelty reward; original GRPO	Adapted efficiency reward
C3L-core	$\hat{y} = S_T$	Volume/coverage rewards, survivor grouping, continuous decay, stratified κ; fixed labels	Complete process without label cache
C3L	$\hat{y} = S_T$	Full method with alternative-label cache	Proposed method

Note. All trained main-table baselines share the data, backbone, SFT initialization, candidates and interaction budget. Named adaptations preserve each method's principal signal and objective.

B.6 Baseline adaptations

Table B.5 summarizes each baseline's output protocol, training signal, and role in the comparison.

Every published method in the main table is retrained under the shared model, the mixed SFT, the data, the candidates, and the tools. A^2Agent-Matched (Kim et al., 2026) uses the locked original annotator to decide which entity counts as discovered. It keeps the first-discovery signal, the Discovery-Commit signal, and terminal F1. The history key $H = 2$ excludes the current action, and the discount is 1 throughout. CodeScout-Matched (Sutawika et al., 2026) keeps GSPO and trains on terminal three-granularity F1. It does not use a reward, specific to a model scale, for stopping on an exact turn. FuseSearch-Seq (Xu et al., 2026) keeps GRPO. Its quality term is adapted to the mean F1 over the three granularities, and the reward is $F(1 + e)$, where e is the mean novelty of the entities returned by each tool relative to the history. The main table calls tools in sequence. The native parallel control is in Appendix A.7. The GRPO and GSPO runs refresh their sampling snapshot every 200 issues, so each iteration's 19,200 rollouts form 12 on-policy rounds of two epochs each under the original clipping ranges, matching C3L's 2,400 optimizer steps per iteration.

Only A^2Agent-Matched shares the imitation-weighted objective with C3L, so the attribution of components uses the same-objective ladder in § 4.4, and the effect of the objective is cross-checked in Appendix A.4. Published CodeScout uses a smaller turn budget and a different training scale. The published quality term of FuseSearch is 0.7 file $+0.3$ function, and that work evaluates the representable subsets of 386 Verified issues and 456 LocBench issues. The adapted names mark those differences in convention. The main table reports only the re-runs under the shared setting.

B.7 METRICS, STATISTICS AND COST ACCOUNTING

Scoring and coverage. At each granularity, precision, recall, and F1 are computed per issue and then macro-averaged. Mean F1 then averages the three granularities with equal weight. The precision of an empty prediction is 1 only when the label is also empty. The recall of an empty label is 1. F1 is 1 when both are empty and 0 when exactly one is empty. On a granularity with an empty label, Free can submit the empty set directly, while Prune must remove every candidate at that granularity. Table A.4 checks the nonempty subset. Complete file coverage is computed only on issues whose reference file set is nonempty. Let $\mathcal{D}$ be the locked evaluation set and $\mathcal{D}_g^+ = \{x : |y_g| > 0\}$. Initial coverage is

$$\mathrm{ICR}_g = \frac{1}{|\mathcal{D}_g^+|} \sum_{x \in \mathcal{D}_g^+} \mathbf{1}[y_g \subseteq C_x^g], \qquad \mathrm{ICR}_{\mathrm{all}} = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbf{1}[\forall g,\ y_g \subseteq C_x^g].$$

The reference ceiling is always computed from the fixed reference label, which is distinct from B_x after the training cache has grown. Output size is the cardinality of the predicted set at each granularity. The empty-prediction rate is the rate of $S_T = \varnothing$, and the P90 of the terminal file count diagnoses over-contraction. PER is restricted to issues whose initial file coverage is already complete, so a retrieval miss is not charged to a search decision.

Replication and pairing. The SD reported in the tables is the sample standard deviation across training seeds. Frozen, untrained, and fixed reference-input rows are a single deterministic value. A one-off audit, and any diagnostic that supplies no variance, receives no error bar. Paired CIs use the per-issue bootstrap described in § 4.1. A two-level bootstrap that resamples seeds and then issues widens the three headline localization intervals to [4.57, 8.45], [4.66, 9.78], and [4.31, 8.12] on Verified, Pro-Python, and LocBench V1, all above zero. A stratified comparison resamples inside its own subset. The same run is reused when a section cites it again. A reported paired difference keeps the rounding of the original full-precision calculation. It is not obtained by subtracting two means that have already been rounded. Table B.6 collects every paired comparison in the main text, together with the per-issue counts of wins, ties, and losses. Table A.9, Figure A.5, and Figure A.23 add the objective cross, the repairer transfer, and the equal-compute comparison.

Table B.6: Paired comparisons and issue-level wins, ties and losses.

Exp.	Bench.	Model	Comparison	Δ (pp)	95% CI	W / T / L
Localization	Verified	4B	C3L - Terminal-Prune	11.81	[10.16, 13.49]	383/25/92
	Verified	4B	C3L - A^{2}Agent-Matched	8.21	[6.70, 9.68]	349/23/128
	Pro-Python	4B	C3L - Terminal-Prune	14.40	[12.29, 16.58]	205/31/30
	Pro-Python	4B	C3L - A^{2}Agent-Matched	7.22	[5.18, 9.29]	173/33/60
	LocBench V1	4B	C3L - A^{2}Agent-Matched	9.00	[7.67, 10.36]	396/53/111
	Verified	8B	C3L - A^{2}Agent-Matched	8.29	[6.75, 9.88]	357/13/130
	Pro-Python	8B	C3L - A^{2}Agent-Matched	7.38	[5.26, 9.55]	166/25/75
	Verified	4B	C3L - FuseSearch-Seq	6.51	[4.94, 8.11]	296/43/161
	LocBench V1	4B	C3L - FuseSearch-Seq	6.21	[4.75, 7.66]	332/47/181
Repair	Verified (all)	4B	C3L - Terminal-Prune	3.87	[2.33, 5.47]	45/445/10
	Verified (all)	4B	C3L - A^{2}Agent-Matched	3.07	[1.53, 4.67]	38/449/13
	Verified (multi-file)	4B	C3L - Terminal-Prune	4.36	[1.56, 7.48]	9/97/1
	Verified (multi-file)	4B	C3L - A^{2}Agent-Matched	3.43	[0.31, 6.54]	8/97/2

Exp.	Bench.	Model	Comparison	Δ (pp)	95% CI	W / T / L
	Verified (all)	8B	C3L - A^2Agent-Matched	4.73	[3.20, 6.27]	46/447/7
	Verified (multi-file)	8B	C3L - A^2Agent-Matched	6.23	[3.12, 9.35]	11/96/0
Components	Verified	4B	Stage 2 - stage 1 (protocol)	1.61	[0.23, 3.02]	258/27/215
	Verified	4B	Stage 3 - stage 2 (volume reward)	1.60	[0.18, 3.02]	254/38/208
	Verified	4B	Stage 4 - stage 3 (coverage loss)	4.29	[2.94, 5.71]	263/47/190
	Verified	4B	Stage 5 - stage 4 (turn advantage)	2.80	[1.48, 4.15]	262/54/184
	Verified	4B	Stage 6 - stage 5 (suffix decay)	1.81	[0.49, 3.12]	243/59/198
Components	Verified	4B	Stage 7 - stage 6 (label cache)	1.31	[-0.09, 2.70]	236/58/206
	Pro-Python	4B	Stage 6 - stage 2 (all process signals)	12.97	[10.79, 15.13]	202/9/55
	Verified	4B	Signature - history grouping (C3L-core)	1.35	[0.19, 2.51]	238/66/196
Reward timing	Verified	4B	Fused advantage: current - terminal	2.11	[0.94, 3.28]	251/96/153
	Pro-Python	4B	Fused advantage: current - terminal	2.29	[0.83, 3.75]	128/57/81
	Verified	4B	Trajectory only: current - terminal (identity)	0.00	[0.00, 0.00]	0/500/0

Note. Localization (§ 4.2), components (§ 4.4) and reward timing (§ 4.5) use Mean F1 differences; repair (§ 4.3) uses resolve-rate differences. Intervals are percentile 95% CIs from 10,000 issue-level bootstrap replicates after averaging paired differences across 3 seeds. Stage numbers refer to Table 3. The final row reuses the same checkpoint and is an identity check; its zero interval is exact. Deltas retain the original full-precision calculation before rounding.

Cost. Localization input tokens accumulate repeated reads of the history, and output tokens include the candidate decisions. We also record the number of tool calls actually made, the number of interaction turns, and end-to-end latency. Per-issue pre-retrieval and maintenance of the candidate set count toward latency. Construction of the repository index is amortized on its own. Training cost is split into demonstrations, rollouts, optimization, calibration generation, and tests. GPU-hours and CPU-hours are reported separately, and wall-clock time is kept. Matching the rollout count matches only the number of samples. An equal-GPU-hour comparison includes only the checkpoints that finished inside the budget. Revisions of data, model, and code, the instance split, the configuration, the seeds, the candidate hash, the checkpoint, and the per-issue results are linked through the run record, so a metric uses one accounting convention across sections.

B.8 Training Procedure Summary

Algorithm 1 summarizes one C3L iteration. The first iteration starts from the shared SFT policy and reference-only label caches; later iterations retain the updated policy and caches. Rollouts follow the action and stopping rules in Appendix B.3. The defaults are $M = 8$ rollouts per issue and $E = 2$ optimization epochs (Appendix B.4).

Algorithm 1 C3L: one sampling-and-training iteration.

Require: Policy π_θ; issues $\mathcal{Q}$; candidates and weights $\{C_x, q_x\}$; label caches $\{\mathcal{Y}_x\}$.
Require: Rollout count M; validation budget k; epochs E; parameters in § B.4.
Ensure: Updated policy π_θ and label caches $\{\mathcal{Y}_x\}$.

Freeze $\theta_{\text{old}} \leftarrow \theta$; initialize the weighted rollout collection $\mathcal{B} \leftarrow \varnothing$.
for each issue $x \in \mathcal{Q}$ **do**
  Sampling. Draw M rollouts under $\pi_{\theta_{\text{old}}}$, indexed by $\mathcal{I}_x$.
  for each rollout $i \in \mathcal{I}_x$ **do**
    $S_{i,0} \leftarrow C_x$; $t \leftarrow 0$.
    while the agent has not stopped and interaction budget remains **do**
      $t \leftarrow t + 1$; call one read-only tool and observe its result.
      Declare $(X_{i,t}, K_{i,t})$.
      $S_{i,t} \leftarrow \text{close}\big((S_{i,t-1} \setminus X_{i,t}) \cap K_{i,t}\big)$. {§ 3.1}
    end while
    $T_i \leftarrow t$; $\hat{y}_i \leftarrow S_{i,T_i}$.
  end for
  Calibration. Using the incoming $\mathcal{Y}_x$, collect scopes with $B_x^f(S_{i,T_i}) < 1$.
  Deduplicate scopes and rank by decreasing $B_x(S_{i,T_i})$ under the same $\mathcal{Y}_x$.
  Validate at most k ranked scopes with the frozen repairer, subject to the cache cap. {§ B.5}
  Add canonical diff labels from patches passing scope and F2P/P2P checks; freeze $\mathcal{Y}_x$.
  Recompute. Under this $\mathcal{Y}_x$, recompute R_x, B_x, B_x^f, and all $r_{i,t}$ for every $i \in \mathcal{I}_x$. {§ 3.2}
  $J_i \leftarrow \sum_{t=1}^{T_i} r_{i,t}$; $G_{i,t} \leftarrow \sum_{s=t}^{T_i} r_{i,s}$, for all $i \in \mathcal{I}_x$ and turns t.
  $d_{i,t} \leftarrow 1$ if $B_x^f(S_{i,0}) = 0$, else $B_x^f(S_{i,t-1})/B_x^f(S_{i,0})$.
  Advantages. Compute A_i^{traj} by standardizing J_i within $\mathcal{I}_x$. {§ 3.3}
  Form $\mathcal{H}_{i,t}$ by issue, $\sigma(S_{i,t-1})$, and the pre-action remaining-budget bin.
  Compute $\hat{A}_{i,t}$ by standardizing $G_{i,t}$ within $\mathcal{H}_{i,t}$. {§ 3.3}
  Set only $\hat{A}_{i,t} \leftarrow 0$ if the group has fewer than two independent rollouts or return SD $\leq \varepsilon_g$.
  $A_{i,t} \leftarrow \omega_0 A_i^{\text{traj}} + \omega_1 \hat{A}_{i,t}$.
  $w_{i,t} \leftarrow d_{i,t}\,\text{clip}\big(\exp(A_{i,t}/\tau), w_{\min}, w_{\max}\big)$.
  Append every rollout $i \in \mathcal{I}_x$ with its turn weights $\{w_{i,t}\}_{t=1}^{T_i}$ to $\mathcal{B}$.
end for
Update. Optimize θ for E epochs on $\mathcal{B}$ using $\mathcal{L}(\theta)$ in § 3.3 and AdamW (§ B.4).
return π_θ and $\{\mathcal{Y}_x\}$.

Calibration precedes reward and advantage computation for every trajectory of the issue, including those whose scopes were not validated. Candidate weights q_x remain fixed throughout. Optimization uses $N = |\mathcal{B}|$ trajectories, masks all non-agent tokens, normalizes each trajectory by $|\mathcal{V}_i|$, and treats rewards, advantages, and weights as stop-gradient. At inference, the learned policy runs a single greedy trajectory through the sampling loop and returns its final survivor set.