BRANCHAID: PRICING SEARCH AGAINST TRAJECTORY RECOVERY IN REPOSITORY AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

On 300 SWE-bench Verified issues, a repository agent solves more tasks by selectively repairing its live trajectory than by spending the same marginal credit on another search branch: GPT-4o moves from 30.2% to 31.7% solved while its solved-task median cost falls from $4.73 to $3.92 and superseded build/test invocations fall from 11.0 to 8.5; Claude 3.5 Sonnet moves from 33.1% to 34.8%, $4.41 to $3.64, and 10.4 to 7.8. The *recovery exchange* makes this allocation measurable through paired end-to-end runs with identical backbones, tools, 40-step and $15 caps, and event-level credit accounting; Extra-Search spends each released credit on exploration, while CostRule shares the recovery action menu and observable state. BRANCHAID implements the recovery policy with a heterogeneous graph of repository execution, calibrated operational hazards, action-conditioned terminal value, cost-aware ranking, and abstention. Across 900 matched issue–seed pairs per backbone, issue-clustered solve intervals favor BRANCHAID over both controls, and every paired cost and superseded-CI interval favors BRANCHAID. The gain concentrates on pre-defined hard issues—3.3 points for GPT-4o and 3.1 for Claude—while easy and mid-band outcomes remain statistically aligned, locating the measured advantage at long or CI-intensive trajectories.

1 INTRODUCTION

Consider a repository agent immediately after a failed narrow test. Its current diff, command history, retrieved symbols, and remaining budget may already identify a correctable mistake, yet the default test-time scaling move is often to generate another hypothesis. Search and correction then differ in two ways at once: what they do and how much additional compute they receive. Repository systems provide increasingly capable search, editing, and execution interfaces (Jimenez et al., 2024; Yang et al., 2024; Zhang et al., 2024; Xia et al., 2024; Wang et al., 2025), while trajectory search expands, ranks, or resumes candidate paths (Antoniades et al., 2025; Ding & Zhang, 2026). What remains unresolved is whether another branch is still the best purchase once the live repository state exposes a specific failure.

The central empirical result is a joint success–efficiency improvement at equal caps. Across 900 matched issue–seed pairs per backbone, BRANCHAID raises GPT-4o solve rate from 30.2% to 31.7% relative to Extra-Search while reducing solved-task median cost by 17.1% and superseded continuous-integration (CI) work by 22.7%. Claude 3.5 Sonnet rises from 33.1% to 34.8%, with 17.5% lower solved-task cost and 25.0% less superseded CI. Exact paired contrasts remain positive against a deterministic controller that observes the same state and chooses from the same recovery menu. The larger 3.3- and 3.1-point gains on pre-defined hard issues identify where the aggregate improvement originates.

These comparisons define the *recovery exchange*. Each end-to-end run fixes the issue, repository snapshot, backbone, prompt, tools, verifier, seed, and 40-step/$15 caps. At a committed tool-result boundary, recovery may abstain or buy targeted retrieval, a static check, an early narrow test, patch decomposition, or build deferral; Extra-Search receives the corresponding event credit for exploration. The primary estimand is the terminal difference between these matched policies, so it includes the downstream trajectory induced by an earlier allocation rather than treating post-action states as if they remained identical.

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.

BRANCHAID supplies the learned side of the exchange. A heterogeneous graph represents files, symbols, tests, diffs, commands, and remaining budget. Calibrated heads identify five operational hazards, an action-conditioned value model estimates improvement over abstention, and a cost-normalized gate executes the best supported correction. This decomposition is technically consequential: risk identifies exposure, value distinguishes among affordable repairs, and abstention leaves productive prefixes undisturbed. Prefix-integrity perturbations, supported action-family estimates, and component ablations connect the end-to-end gain to this selective mechanism.

The paper makes three linked contributions:

- The recovery exchange supplies an end-to-end matched-credit estimand and event ledger for comparing search with concrete trajectory correction without increasing the agent's caps.
- BRANCHAID couples heterogeneous live execution state, calibrated multi-label hazards, action-conditioned advantage over abstention, cost-aware ranking, and selective execution.
- Equal-credit search and same-menu rule controls, paired issue-level inference, integrity stress tests, and a pre-defined regime analysis show simultaneous gains in resolution, spend, and discarded CI on long or CI-intensive trajectories.

2 THE RECOVERY EXCHANGE ACROSS COMMITTED DECISIONS

A repository agent executes steps $t = 1, \ldots, T$ and observes prefix state s_t: the current repository snapshot, retrieved context, accumulated diffs, command history, completed build/test outputs, and remaining budget b_t. Its backbone action $a_t \in \mathcal{A}$ incurs metered cost $c(a_t)$. We study five non-exclusive hazards $\mathcal{H} = \{h_1, \ldots, h_5\}$: focus on the wrong file, an under-tested patch, an environment mismatch, dependency breakage, and over-wide edit scope. The recovery menu $\mathcal{I} = \{i_0, \ldots, i_5\}$ contains abstain, targeted retrieval, static check, early narrow test, patch decomposition, and build deferral.

An intervention opportunity occurs after an external tool result is committed to the trace and before the next backbone call. The harness meters model input/output tokens at provider prices, retrieval and editing execution, CI build/test time, and controller GPU time into one per-task dollar ledger. For action i_j, $\hat{c}_t(i_j)$ is the router's expected charge and $\bar{c}_t(i_j)$ is the harness reservation bound; dispatch requires $\bar{c}_t(i_j) \leq b_t$, and unused reservation is released after the realized charge is recorded. The realized charge becomes the event credit that Extra-Search may spend on retrieval and exploration until its next call would exceed the released amount. Abstention releases no credit. Residual credit remains within the task, never transfers across tasks, and the reservation plus the 40-step pre-dispatch check enforces both caps.

Matching operates at two levels. All policies start from the same issue–seed state and use common prompts, tools, verifier, and caps. Event credits match the marginal spend rule, but earlier choices may change later prefixes and even the number of later opportunities. No synthetic opportunity is created after an arm terminates. This design intentionally evaluates the deployable end-to-end policy effect; the separate randomized decision episodes in Section 4 estimate action advantage at cloned committed prefixes.

The shared objective is

$$\max_{\pi_I} \mathbb{E}\left[Y \;\middle|\; \sum_{t=1}^{T} (c(a_t) + c(\pi_I(s_t, b_t))) \leq B \right], \qquad Y = \mathbf{1}[\text{verifier-correct patch}], \tag{1}$$

where π_I observes only the committed prefix. Let π_S denote Extra-Search under the reconciled credit rule and let q index matched issue–seed pairs. The policy-level recovery-exchange gain is

$$\Gamma_Y(\pi_I; \pi_S) = \mathbb{E}_q[Y_q(\pi_I) - Y_q(\pi_S)]\,. \tag{2}$$

Equation 1 fixes feasibility; Equation 2 measures the terminal consequence of changing the allocation policy while holding the surrounding stack and caps fixed. Paired cost and superseded-CI differences supply the resource coordinates of the same exchange. Positive solve gain with negative resource intervals means that recovery converts the available budget more effectively than continued search in the evaluated policy trajectories.

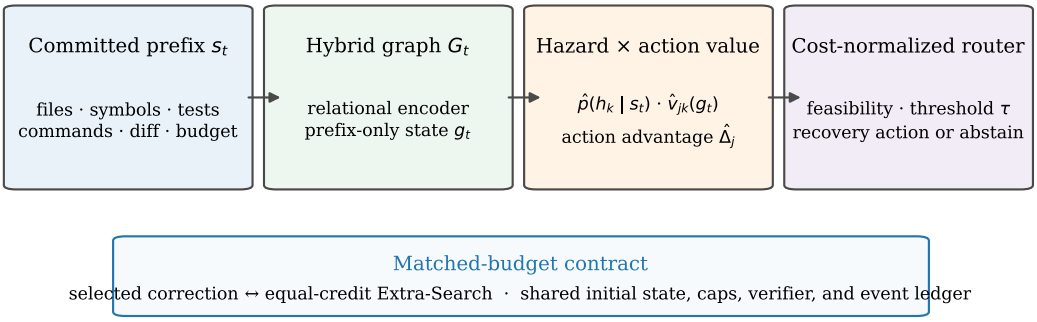


Figure 1: BRANCHAID's recovery-exchange pipeline. Runtime features end at the committed prefix; calibrated hazard and action-value estimates rank feasible corrections per unit cost, and thresholding selects recovery or abstention. The ledger gives Extra-Search the released event credit under the same initial state, caps, tools, and verifier.

3 METHOD: BRANCHAID HAZARD-TO-ACTION RECOVERY ROUTING

BRANCHAID implements the recovery exchange with a hazard-to-action decomposition. It reconstructs each committed prefix as a hybrid graph $G_t = (V_t, E_t)$. Nodes represent `file`, `symbol`, `test`, `diff`, `command`, and `budget` entities; import, call, coverage, edit-order, and retrieval-order edges preserve executable and temporal structure. A three-layer relational graph attention network (Busbridge et al., 2019) combines frozen CodeBERT code features (Feng et al., 2020) with a lightweight transformer over diff nodes to produce a 256-dimensional state $\mathbf{g}_t$. Figure 1 summarizes the complete prefix-to-action path and its matched search arm.

Five two-layer, 128-unit GELU heads estimate prefix hazards:

$$\hat{p}_t^{(k)} = \sigma(W_k \,\mathrm{MLP}_k(\mathbf{g}_t) + b_k)\,, \qquad k = 1, \ldots, 5. \tag{3}$$

Training minimizes binary cross-entropy plus an expected calibration error (ECE) penalty,

$$\mathcal{L}_{\text{hazard}} = \sum_{k=1}^{5} \left[-y_t^{(k)} \log \hat{p}_t^{(k)} - (1 - y_t^{(k)}) \log(1 - \hat{p}_t^{(k)}) + \lambda_{\text{cal}} \mathrm{ECE}_k \right], \tag{4}$$

where ECE_k is a differentiable soft-binned calibration surrogate and $\lambda_{\text{cal}} = 0.1$. Per-head temperature scaling is then fit on held-out prefixes (Platt, 1999; Guo et al., 2017; Kumar et al., 2019). Future trace evidence supplies the wrong-file, testing, environment, dependency, and scope training labels; every feature presented to the deployed heads ends at the committed decision boundary.

Let $\mathbf{x}_t = (\mathbf{g}_t, \hat{\mathbf{p}}_t, b_t)$ collect graph state, calibrated hazards, and remaining budget. The value target for action i_j is its conditional terminal advantage over abstention,

$$\Delta_j(\mathbf{x}_t) = \mathbb{E}[Y \mid A_t = i_j, \mathbf{x}_t] - \mathbb{E}[Y \mid A_t = i_0, \mathbf{x}_t]. \tag{5}$$

A bilinear interaction among $\mathbf{g}_t$, intervention embeddings, and hazard embeddings yields additive components $\hat{v}_{jk}$, whose calibrated aggregate is $\hat{\Delta}_j(\mathbf{x}_t) = \sum_k \hat{p}_t^{(k)} \hat{v}_{jk}(\mathbf{g}_t)$. The components attribute the score across overlapping hazards; the aggregate is trained directly against the action-versus-abstention target in Equation 5. The router first selects the best feasible non-abstain action,

$$j^\star = \operatorname*{argmax}_{\substack{j \in \{1,\ldots,5\} \\ \bar{c}_t(i_j) \le b_t}} \frac{\hat{\Delta}_j(\mathbf{x}_t)}{\hat{c}_t(i_j) + \epsilon}, \qquad \epsilon = 10^{-4}, \tag{6}$$

If no non-abstain action is feasible, the router returns i_0; otherwise it applies abstention to the selected action:

$$\pi_I(s_t, b_t) = \begin{cases} i_{j^\star}, & \hat{\Delta}_{j^\star}(\mathbf{x}_t) \ge \tau, \\ i_0, & \text{otherwise}, \end{cases} \qquad \tau = 0.15. \tag{7}$$

Table 1: Partition and timing account. Issue-disjoint means no issue identity crosses the listed stage and the 300-issue main evaluation; repository-disjoint applies to shifted transfer evaluation.

Stage	Unit	Separation	Frozen output
Hazard fitting	Step-aligned prefix	Issue-disjoint	Graph encoder and hazard heads
Temperature calibration	Prefix	Issue-disjoint	Per-head temperatures
Randomized value estimation	Decision episode	Issue-disjoint	Action-value model
Main matched evaluation	Issue $\times$ seed	Fresh 300 issues	All policies and caps
Transfer evaluation	Repository/task $\times$ seed	Repository-disjoint	Router; temperatures recalibrated

The threshold is selected once on the calibration partition for verifier success under metered cost. This construction converts prefix risk into action-specific advantage: hazard heads locate recoverable exposure, the value interaction distinguishes useful corrections, and the threshold reserves credit when no action has sufficient conditional support.

4 Training, Timing, and Estimation

Training examples are step-aligned logged prefixes. Hazard fitting, temperature calibration, randomized action-value estimation, main matched runs, and transfer evaluation use separate issue-level partitions; the transfer partition is additionally repository-disjoint. Main-evaluation issues never enter fitting or threshold selection, and randomized value episodes never enter headline tables. Thresholds and CostRule symptom cutoffs are frozen after calibration.

Each randomized decision episode clones a committed repository snapshot and model context, samples one feasible menu action from a logged behavior policy with recorded propensity, and then lets the unchanged backbone continue to the terminal verifier. The abstention branch is cloned from the same prefix. Repeated episodes may share an issue and terminal label, so every episode from an issue remains in the same fitting fold and all reported uncertainty clusters at the issue. This local fork supplies Equation 5; the main table instead measures the complete end-to-end policy in Equation 2.

The 300-issue study population comprises the SWE-bench Verified issues in the 12 Python repositories supported by the shared live build/test harness. Hard/easy/mid membership is frozen before the evaluated policy runs from a separate Extra-Search reference execution and then held fixed across both backbones and all evaluation seeds: hard means more than 10 steps or more than \$0.50 per CI invocation, easy means at most 5 steps and at most \$0.20, and mid is the remainder. The reference execution is not counted among the three evaluation seeds. Table 1 records the timing and separation constraints.

Action value uses issue-level cross-fitted doubly robust pseudo-outcomes (Dudík et al., 2011; Precup et al., 2000; Thomas & Brunskill, 2016). For each supported action, nuisance outcome and propensity models are trained without issue $q(t)$:

$$\tilde{\mu}_{t,j} = \hat{\mu}_j^{(-q(t))}(\mathbf{x}_t) + \frac{\mathbf{1}[A_t = i_j]}{\hat{e}_j^{(-q(t))}(\mathbf{x}_t)} \left(Y_t - \hat{\mu}_j^{(-q(t))}(\mathbf{x}_t) \right), \qquad \tilde{\Delta}_{t,j} = \tilde{\mu}_{t,j} - \tilde{\mu}_{t,0}. \tag{8}$$

Inverse-propensity factors are clipped at 10, and the bilinear value model minimizes squared error to $\tilde{\Delta}_{t,j}$ over actions with logged propensity at least 0.08. Appendix Table A.4 reports the resulting support, effective sample size, action-family averages, and issue-clustered intervals. Those family averages summarize broad supported episodes; the online gate uses the conditional $\hat{\Delta}_j(\mathbf{x}_t)$. A stratified two-annotator audit independently validates the operational hazard labels (Appendix Table A.6).

Table 2: Operational policy comparison. All rows use the same backbone, tool interface, opportunity timing, verifier, and pre-dispatch cap checks.

Policy	Observable state	Output/menu	Trigger/tuning	Spend rule
No Predictor	Backbone prefix	Backbone action	None	Base ledger
Fixed-Budget	Prefix, budget	Fixed recovery slot	Calibration split	Charged event credit
Self-Refine	Prefix, feedback	Revision feedback	Published-style trigger	Charged event credit
Static-Analysis	Prefix, diff	Static check	Tool-valid diff	Charged event credit
Lookahead	Prefix, candidates	One-step rollout	Calibration split	Charged event credit
Extra-Search	Prefix, budget	Retrieval/search	Every released credit	Matched event credit
CostRule	Identical graph fields	Full recovery menu	Held-out symptom rules	Matched event credit
BRANCHAID	Identical graph fields	Full recovery menu	Value threshold τ	Matched event credit

5 EXPERIMENTAL DESIGN

The experiments ask six questions: Does recovery beat equal-credit search? Does learned action value beat a strong same-menu rule? Which trajectories benefit? Which components create the gain? Does the prefix-only mechanism survive temporal-proxy stress? How does repository shift move the operating region? The benchmark is 300 SWE-bench Verified issues (Jimenez et al., 2024) from 12 Python repositories with live build and test pipelines. GPT-4o (OpenAI, 2024) and Claude 3.5 Sonnet (Anthropic, 2024) run as ReAct-style backbones (Yao et al., 2023) with identical prompts, retrieval, editing, test tools, verifiers, 40-step limits, and $15 caps.

Comparators are No Predictor, Fixed-Budget, Self-Refine (Madaan et al., 2023), Static-Analysis, Lookahead Rollout, Extra-Search, and CostRule. Extra-Search spends released credit on additional repository queries and candidate exploration, stopping before the next call would exceed that event credit. CostRule observes the same graph fields as BRANCHAID, abstains when no rule fires, and uses the same five non-abstain actions: repeated irrelevant retrieval routes targeted lookup; syntax/type evidence routes static checking; a coherent untested diff routes a narrow test; excessive diff dispersion routes decomposition; and expensive low-information CI routes deferral. If several symptoms fire, the rule with the largest calibration-set cost-adjusted success is chosen; exact ties follow the fixed menu order. Rule cutoffs and priority are frozen on the calibration partition. Table 2 records state, menu, trigger, tuning, and spend for every comparator.

Training uses $8\times$A100 80GB GPUs with CUDA 12.4 and PyTorch 2.3.1; online routing uses one A100. Each backbone–policy pair runs end-to-end for three seeds. Solve rate is final verifier success. Cost is the median among solved tasks and is interpreted alongside failure-inclusive mean spend and resolved issues per $100. Wasted CI is the mean number of build/test invocations superseded by a later edit; it measures discarded work rather than total test quality. Intervention precision is the fraction of non-abstain choices followed by the intended hazard-resolution event.

The pre-specified primary contrast is BRANCHAID versus Extra-Search on solve rate. The same-menu CostRule contrast is confirmatory; regime, component, calibration, and transfer analyses diagnose mechanism and scope. Primary inference uses 10,000 issue-clustered bootstrap resamples over all 900 task-seed pairs; ties are identical binary verifier outcomes. Cost intervals use jointly solved pairs, and all-run spend/yield retains failures. Seed standard errors in Table 3 describe rerun variability, while paired intervals in Table 4 support the policy comparisons.

Table 3: Primary recovery-exchange endpoint on 300 SWE-bench Verified issues. Solve is final verifier success; cost is the solved-task median; wasted CI counts superseded build/test invocations per run; intervention precision conditions on non-abstain choices. Values after ± are standard errors over three reruns.

Backbone	Policy	Solve (%)	Cost ($)	Wasted CI	Intv. precision
GPT-4o	No Predictor	27.3 ± 1.1	4.82 ± 0.21	12.4 ± 0.8	—
	Fixed-Budget	28.1 ± 0.9	4.51 ± 0.18	11.2 ± 0.7	—
	Self-Refine	28.7 ± 1.0	4.38 ± 0.19	10.8 ± 0.6	0.41
	Static-Analysis	29.4 ± 0.8	4.14 ± 0.17	9.7 ± 0.5	0.52
	Lookahead	29.1 ± 1.1	4.61 ± 0.22	10.3 ± 0.7	0.47
	Extra-Search	30.2 ± 0.9	4.73 ± 0.20	11.0 ± 0.6	—
	CostRule	30.5 ± 0.8	4.18 ± 0.17	9.2 ± 0.5	0.61
	BRANCHAID	31.7 ± 0.7	3.92 ± 0.15	8.5 ± 0.4	0.68
Claude 3.5	No Predictor	30.1 ± 1.0	4.47 ± 0.19	11.1 ± 0.7	—
	Fixed-Budget	31.0 ± 0.8	4.18 ± 0.17	10.3 ± 0.6	—
	Self-Refine	31.5 ± 0.9	4.05 ± 0.18	9.9 ± 0.5	0.44
	Static-Analysis	32.3 ± 0.7	3.87 ± 0.16	9.1 ± 0.5	0.55
	Lookahead	31.9 ± 1.0	4.29 ± 0.20	9.7 ± 0.6	0.49
	Extra-Search	33.1 ± 0.8	4.41 ± 0.19	10.4 ± 0.6	—
	CostRule	33.6 ± 0.7	3.90 ± 0.15	8.5 ± 0.4	0.64
	BRANCHAID	34.8 ± 0.6	3.64 ± 0.14	7.8 ± 0.4	0.71

Table 4: Paired recovery-exchange evidence for BRANCHAID over the two strongest comparators. W/T/L counts cover 900 matched task-seed pairs. Intervals are 95% issue-clustered bootstrap intervals; cost uses jointly solved pairs.

Backbone	Comparator	W/T/L	Solve Δ (pp)	Cost Δ ($)	Wasted-CI Δ
GPT-4o	Extra-Search	34/845/21	1.4 [0.3, 2.8]	$[-1.02, -0.57]$	$[-2.9, -2.1]$
	CostRule	19/873/8	1.2 [0.1, 2.3]	$[-0.44, -0.08]$	$[-1.0, -0.4]$
Claude 3.5	Extra-Search	35/845/20	1.7 [0.4, 2.9]	$[-0.98, -0.54]$	$[-3.0, -2.2]$
	CostRule	20/871/9	1.2 [0.1, 2.4]	$[-0.44, -0.07]$	$[-1.0, -0.4]$

6 RECOVERY EXCHANGE RESULTS

Table 3 places BRANCHAID in the preferred corner of the exchange: more solved issues with less resource waste. Against GPT-4o Extra-Search, the exact paired contrast is 1.4 points while solved-task median cost falls by 17.1% and wasted CI by 22.7%; against CostRule, solve rate rises by 1.2 points while cost falls by 6.2% and wasted CI by 7.6%. Claude reproduces the joint result: a 1.7-point solve gain, 17.5% lower solved-task cost, and 25.0% less wasted CI than Extra-Search, plus a 1.2-point gain over CostRule with lower resource use. Independently rounded solve-rate cells in Table 3 may differ by 0.1 point from contrasts computed directly from the binary pairs.

The Extra-Search rows directly estimate Equation 2; the CostRule rows control prefix state, action menu, and spend rule when assessing the full learned router. Both backbones retain positive solve intervals against equal-credit search and the same-menu rule, and every cost and wasted-CI interval lies below zero. CostRule's improvement over weaker controls confirms that disciplined symptom–action mapping is competitive; BRANCHAID's remaining advantage is the increment associated with calibrated, prefix-specific action ranking.

The activation map explains where recovery earns its credit. Exact paired outcomes improve hard-issue solve rate by 3.3 points on GPT-4o and 3.1 points on Claude, with intervals above zero. Easy and mid intervals overlap zero, while their point estimates remain close through abstention. The aggregate gain therefore comes from redirecting expensive detours while leaving productive trajectories with the backbone.

Table 5: Pre-defined activation map for the recovery exchange. Each issue has three matched seeds; n is the issue count. The final column is the paired 95% interval for BRANCHAID minus Extra-Search. Policy-rate cells are rounded independently; Figure 2 uses exact paired point contrasts.

Backbone	Regime	n	Extra-Search	CostRule	BRANCHAID	Δ 95% CI (pp)
GPT-4o	Hard	140	20.3	21.1	23.7	[1.0, 5.8]
	Easy	80	48.7	48.4	48.3	[-2.1, 1.4]
	Mid	80	29.0	29.1	29.0	[-1.7, 1.7]
Claude 3.5	Hard	140	23.1	24.2	26.2	[0.8, 5.4]
	Easy	80	51.5	51.4	51.2	[-2.0, 1.4]
	Mid	80	32.2	32.3	33.4	[-0.7, 3.1]

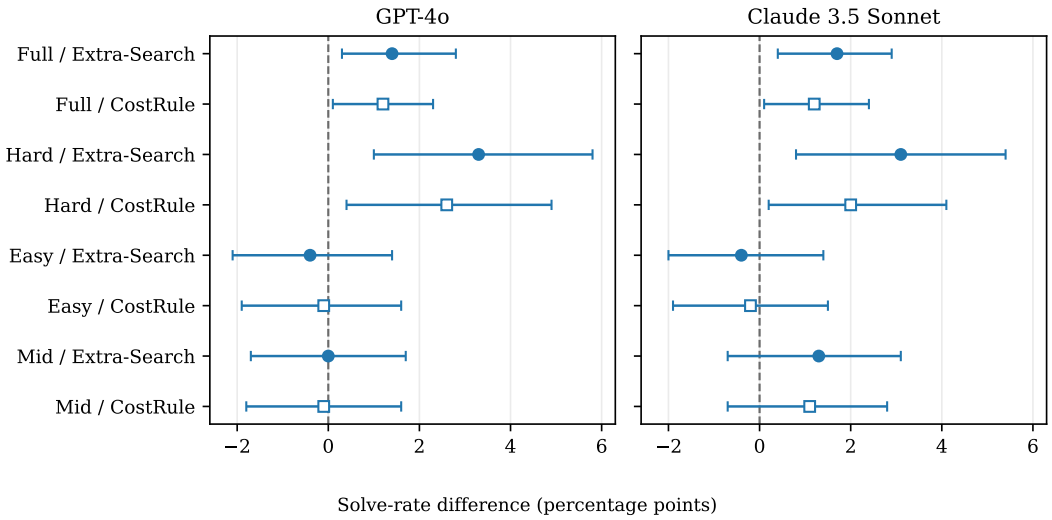


Figure 2: Solve-rate differences for BRANCHAID against Extra-Search and CostRule on full, hard, easy, and mid regimes. Bars are paired 95% issue-clustered bootstrap intervals; the vertical line marks equal performance. Both backbones show resolved full-set and hard-regime gains with neutral easy/mid behavior.

Table 6: Prefix-integrity stress test on 4320 held-out GPT-4o prefixes and the 300-issue endpoint. AUC is hazard ROC-AUC; the future-signal row is an invalid positive control used only to verify audit sensitivity.

Feature pipeline	AUC	ECE	Solve (%)	Solve Δ 95% CI (pp)
Deployed prefix	0.716	0.042	31.7	Reference
Proxy-scrubbed	0.709	0.046	31.5	[-0.9, 0.5]
Order/timestamp perturbation	0.701	0.050	31.2	[-1.3, 0.2]
Invalid future signal	0.889	0.028	35.6	[2.3, 5.6]

Removing plausible temporal proxies or perturbing within-prefix order leaves endpoint performance statistically aligned with the deployed pipeline. Deliberately injecting forbidden future information instead produces a conspicuous AUC, calibration, and solve-rate jump. The contrast shows that the audit detects leakage when present and that BRANCHAID's measured advantage is stable to the principal prefix-only stressors.

The supported-subset analysis uses 247 decision episodes, minimum propensity 0.08, weight cap 10, and effective sample size 182. Direct, inverse-propensity-weighted, and doubly robust estimates agree in ranking targeted retrieval first, with 2.8-point lift [1.1, 4.5], and early narrow testing second, with 2.1-point lift [0.6, 3.7]. Their positive intervals and complementary savings identify the two actions that most directly stop another search or CI branch; Appendix Table A.4 gives all families.

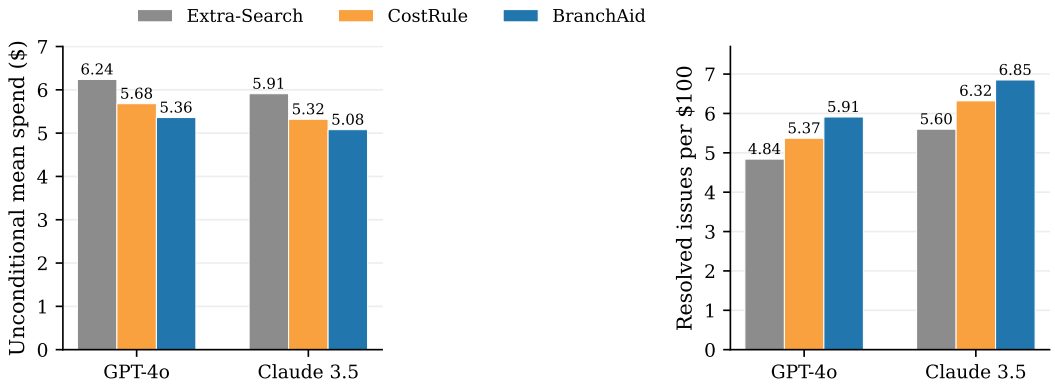


Figure 3: Controller-inclusive all-run efficiency. Relative to equal-spend search and the same-menu rule, BRANCHAID lowers unconditional mean spend and increases resolved issues per $100 on both backbones. Appendix Table A.8 reports the corresponding values and routing latency.

Component ablations reinforce this mechanism. Removing action value costs 1.6 solve points, removing randomized support costs 1.9 points, and decoupling action choice from prefix state costs 2.8 points (Appendix Table A.5). Per-hazard calibration reaches 0.042 macro ECE and 0.184 Brier score, with intervention precision ranging from 0.61 to 0.74 (Appendix Table A.3). An independent audit obtains macro precision 0.82 and recall 0.77 (Appendix Table A.6), aligning calibrated risk, supported action value, and selective execution with adjudicated operational states.

Controller-inclusive accounting preserves the joint gain (Appendix Table A.8). On one A100 80GB GPU, routing adds 0.36 seconds median and 0.74 seconds at p95 per decision, only 1.8% of end-to-end wall time. GPT-4o unconditional spend falls from 6.24 to 5.36, while yield rises from 4.84 to 5.91 resolved issues per $100; Claude shows the same direction.

7 RELATED WORK

Repository-agent systems provide increasingly capable interfaces for search, editing, and execution (Yang et al., 2024; Zhang et al., 2024; Xia et al., 2024; Tao et al., 2024; Wang et al., 2025). SWE-Search expands trajectories with Monte Carlo tree search and value-guided refinement (Antoniades et al., 2025); SWE-Replay resumes archived trajectories at selected branch points (Ding & Zhang, 2026); and SWE-Router escalates the model from partial software-engineering trajectories (Son et al., 2026). These methods allocate additional search or capability. The recovery exchange instead prices a fixed-backbone, tool-level correction against the search that its realized charge could buy, making Extra-Search the direct allocation control.

Several recent systems intervene in coding trajectories. Wink detects production misbehaviors and injects asynchronous corrective guidance (Nanda et al., 2026). Process-centric trajectory graphs support online diagnostics and rollback (Liu et al., 2026a), while LivePlan couples deterministic drift rules to LLM-generated advice (Liu et al., 2026b). FailFast–RestartSmart terminates likely failures and starts a fresh rollout with an optional diff overlay (Wang et al., 2026). CodeRescue routes a failed coding attempt among reflection, replanning, and model escalation and calibrates its post-failure cost frontier (He et al., 2026). BRANCHAID targets a different decision granularity: repeated committed prefixes inside one live repository trajectory, a fixed backbone, and concrete retrieval/check/test/diff/build actions. Its same-menu CostRule control separates learned state–action value from deterministic symptom routing, and its equal-credit search control separates recovery from simply spending more.

Self-Refine and Reflexion revise behavior through generated feedback (Madaan et al., 2023; Shinn et al., 2023), SWE-PRM supplies trajectory-level corrective feedback (Gandhi et al., 2025), and budget-aware tool use exposes remaining resources during test-time scaling (Liu et al., 2025). Warning and fault-localization models estimate risk without valuing a concrete recovery purchase (Kamei et al., 2013; Panichella et al., 2016; Yang et al., 2015; Wong et al., 2016; Li et al., 2019; Huang et al., 2026);

calibration and failure-prediction methods make such scores operationally reliable (Lakshminarayanan et al., 2017; Ovadia et al., 2019; Kumar et al., 2019; Corbière et al., 2019). BRANCHAID joins calibrated prefix hazards to action-conditioned verifier advantage. CI optimization acts on test schedules, commits, or build outcomes (Elbaum et al., 2014; Bagherzadeh et al., 2022; Panichella et al., 2016; Islam & Zibran, 2017); recovery routing acts before the next search branch or CI cycle is purchased.

8 DISCUSSION

The mechanism is consistent across endpoint, regime, and intervention-family evidence. Targeted retrieval repairs wrong-file branches before they accumulate irrelevant context; early narrow testing exposes local patch errors before a full build amplifies their cost. Patch decomposition helps when diff scope becomes the dominant risk, while abstention leaves short productive trajectories with the backbone. The value head matters because a calibrated hazard locates exposure, whereas the action-conditioned target identifies the correction most likely to improve terminal success enough to repay its execution cost. The ablations in Appendix Table A.5 recover this hierarchy.

The matched policy comparison also changes what an efficiency claim must show. Gains that simultaneously alter prompts, calls, caps, and action space mix allocation with additional opportunity. Here the backbone and caps remain fixed, the search comparator receives the released event credit, and the same-menu comparator receives the recovery interface. Higher resolution therefore appears together with lower solved-task cost, lower failure-inclusive spend, higher resolved-issue yield, and less discarded CI rather than being purchased by an unreported compute increase.

The evidence defines a concrete operating envelope: prefix-visible hazards addressed by five within-harness actions, 40-step and $15 caps, and Python repositories with live verification. Solved-only median cost is therefore read with the all-run spend/yield analysis, and superseded CI quantifies discarded work rather than total CI volume or test quality. Lightweight recalibration improves ECE in every shifted setting and preserves a positive solve interval on held-out Python; TypeScript, monorepo, and mixed-language intervals overlap zero. Within the supported regime, the hard/easy/mid map and abstention behavior locate recovery's benefit without extending it to every trajectory or ecosystem.

9 CONCLUSION

BRANCHAID tests a concrete alternative to test-time search: spend the next available credit on the live repository state when a supported correction has sufficient terminal value. Across two backbones, this policy beats both equal-credit Extra-Search and the same-menu CostRule with positive paired intervals while reducing conditional cost, failure-inclusive spend, and superseded CI. On GPT-4o, solve rate rises from 30.2% to 31.7%, solved-task cost falls by 17.1%, and wasted CI falls by 22.7%; Claude shows the same joint direction. The paired gain is concentrated on pre-defined hard issues (3.3 points for GPT-4o and 3.1 for Claude), while easy and mid-band intervals overlap zero. Under the evaluated Python, 40-step, and $15 regime, these outcomes establish that action-valued recovery can outperform another search branch without additional budget.

REPRODUCIBILITY STATEMENT

Sections 2–5 and the appendix specify the estimand, policy, observable state, accounting, architecture, partitions, baselines, metrics, hardware, seeds, inference, and supporting diagnostics.

AI USE STATEMENT

Generative AI tools assisted manuscript preparation, including prose, organization, presentation, and this submission-preparation pass. This documented use did not provide the reported empirical evidence; the authors take responsibility for the final content.

REFERENCES

Anthropic. Claude 3.5 sonnet model card addendum. *Technical Report*, 2024. URL https://www-cdn.anthropic.com/f2986af8d052f26236f6251da62d16172cfabd6e/claude-3-model-card.pdf.

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Yang Wang. SWE-Search: Enhancing software agents with monte carlo tree search and iterative refinement. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G7sIFXugTX.

Mojtaba Bagherzadeh, Nafiseh Kahani, and Lionel Briand. Reinforcement learning for test case prioritization. *IEEE Transactions on Software Engineering*, 48(8):2836–2856, 2022. doi: 10.1109/TSE.2021.3070549.

Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019. doi: 10.48550/arXiv.1904.05811.

Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/757f843a169cc678064d9530d12a1881-Abstract.html.

Yifeng Ding and Lingming Zhang. SWE-Replay: Efficient test-time scaling for software engineering agents. *arXiv preprint arXiv:2601.22129*, 2026. doi: 10.48550/arXiv.2601.22129.

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011. doi: 10.48550/arXiv.1103.4601.

Sebastian Elbaum, Gregg Rothermel, and John Penix. Techniques for improving regression testing in continuous integration development environments. In *ACM SIGSOFT Symposium on Foundations of Software Engineering*, pp. 235–245, 2014. doi: 10.1145/2635868.2635910.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. Codebert: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1536–1547, 2020. doi: 10.18653/v1/2020.findings-emnlp.139.

Shubham Gandhi, Jason Tsay, Jatin Ganhotra, Kiran Kate, and Yara Rizk. When agents go astray: Course-correcting SWE agents with PRMs. *arXiv preprint arXiv:2509.02360*, 2025. doi: 10.48550/arXiv.2509.02360.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Qijia He, Jiayi Cheng, Chenqian Le, Rui Wang, Xunmei Liu, Yixian Chen, Jie Mei, Zhihao Wang, Xupeng Chen, Yuhuan Chen, and Tao Wang. Coderescue: Budget-calibrated recovery routing for coding agents. *arXiv preprint arXiv:2607.19338*, 2026. doi: 10.48550/arXiv.2607.19338.

Xinmiao Huang, Jinwei Hu, Rajarshi Roy, Changshun Wu, Yi Dong, and Xiaowei Huang. PrefixGuard: From LLM-agent traces to online failure-warning monitors. *arXiv preprint arXiv:2605.06455*, 2026. doi: 10.48550/arXiv.2605.06455.

Md Rakibul Islam and Minhaz F Zibran. Insights into continuous integration build failures. In *IEEE/ACM International Conference on Mining Software Repositories*, pp. 467–470, 2017. doi: 10.1109/MSR.2017.30.

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 *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi. A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering*, 39(6):757–773, 2013. doi: 10.1109/TSE.2012.70.

Ananya Kumar, Percy Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/f8c0c968632845cd133308b1a494967f-Abstract.html.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

Xia Li, Wei Li, Yuqun Zhang, and Lingming Zhang. Deepfl: Integrating multiple fault diagnosis dimensions for deep fault localization. In *International Symposium on Software Testing and Analysis*, pp. 174–185, 2019. doi: 10.1145/3293882.3330574.

Shuyang Liu, Yang Chen, Rahul Krishna, Saurabh Sinha, Jatin Ganhotra, and Reyhaneh Jabbarvand. Process-centric analysis of agentic software systems. *Proceedings of the ACM on Programming Languages*, 10(OOPSLA1):1961–1988, 2026a. doi: 10.1145/3798271.

Shuyang Liu, Saman Dehghan, Ji Young Kim, Jatin Ganhotra, Martin Hirzel, and Reyhaneh Jabbarvand. Online monitoring and corrective steering of programming agents. *arXiv preprint arXiv:2608.06701*, 2026b. doi: 10.48550/arXiv.2608.06701.

Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-aware tool-use enables effective agent scaling. *arXiv preprint arXiv:2511.17006*, 2025. doi: 10.48550/arXiv.2511.17006.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594, 2023. doi: 10.52202/075280-2019.

Rahul Nanda, Chandra Maddila, Smriti Jha, Euna Mehnaz Khan, Matteo Paltenghi, and Satish Chandra. Wink: Recovering from misbehaviors in coding agents. *arXiv preprint arXiv:2602.17037*, 2026. doi: 10.48550/arXiv.2602.17037.

OpenAI. Gpt-4o system card. *Technical Report*, 2024. URL https://openai.com/index/gpt-4o-system-card/.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/8558cb408c1d76621371888657d2eb1d-Abstract.html.

Annibale Panichella, Carol V Alexandru, Sebastiano Panichella, Alberto Bacchelli, and Harald C Gall. A search-based training algorithm for cost-aware defect prediction. In *Genetic and Evolutionary Computation Conference*, pp. 1077–1084, 2016. doi: 10.1145/2908812.2908938.

John C Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pp. 61–74, 1999. URL https://www.csie.ntu.edu.tw/~cjlin/papers/platt.pdf.

Doina Precup, Richard S Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 759–766, 2000. URL https://www.cs.utexas.edu/~shivaram/readings/b2hd-PrecupSS2000.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377.

Seongho Son, Sangwoong Yoon, Jiahua Tang, Shuhan Wang, Lorenz Wolf, and Ilija Bogunovic. SWE-Router: Routing in multi-turn agentic software engineering tasks. *arXiv preprint arXiv:2607.00053*, 2026. doi: 10.48550/arXiv.2607.00053.

Wei Tao, Yucheng Zhou, Yanlin Wang, Wenqiang Zhang, Hongyu Zhang, and Yu Cheng. Magis: Llm-based multi-agent framework for github issue resolution. In *Advances in Neural Information Processing Systems*, volume 37, pp. 51963–51993, 2024. doi: 10.52202/079017-1647.

Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 2139–2148, 2016. URL https://proceedings.mlr.press/v48/thomasa16.html.

Chenyu Wang, Yunbo Lyu, Junda He, Zhou Yang, Chenxing Zhong, Yaniv Harel, and David Lo. Fail-fast, restart-smart: Early failure prediction and restart for SWE agentic tasks. *arXiv preprint arXiv:2608.03222*, 2026. doi: 10.48550/arXiv.2608.03222.

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 *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

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. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024. doi: 10.48550/arXiv.2407.01489.

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, pp. 50528–50652, 2024. doi: 10.52202/079017-1601.

Xinli Yang, David Lo, Xin Xia, Yun Zhang, and Jianling Sun. Deep learning for just-in-time defect prediction. In *IEEE International Conference on Software Quality, Reliability and Security*, pp. 17–26, 2015. doi: 10.1109/QRS.2015.14.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1592–1604, 2024. doi: 10.1145/3650212.3680384.

A Operational Definitions

Graph state. The harness canonicalizes repository-relative file paths, language-server symbols, test identifiers, diff hunks, and command invocations before node construction. Duplicate entities share a node, while edit and retrieval order remain directed event edges. The relational encoder performs type-specific attention at each layer and attention-pools the final file, symbol, test, diff, and command representations; concatenating that pool with the budget node produces $\mathbf{g}_t$. Missing node types receive a learned null embedding, so the online input dimension is fixed across prefixes.

Hazards and actions. Wrong-file focus marks prefixes concentrated outside the files or symbols implicated by the eventual verifier-relevant repair; under-testing marks edited behavior not exercised by a targeted test before broader validation; environment mismatch distinguishes toolchain or setup failures from patch failures; dependency breakage marks import, API, or package failures introduced by the current edit; and over-wide scope marks changes outside the repair-relevant dependency neighborhood. These future-visible conditions supervise training labels only. At runtime, targeted retrieval issues a repository query anchored to the highest-scoring file/symbol neighborhood; static check runs the configured syntax and type checks on edited units; early narrow test invokes the nearest impacted test target; patch decomposition separates independent diff regions for sequential validation; and build deferral postpones one broad build while preserving the current repository state. Each action returns its ordinary tool output to the backbone.

Decision and diagnostic units. A decision episode is one cloned committed prefix, one sampled feasible action (including abstention), and the unchanged continuation to terminal verification. A hazard-resolution event occurs when the next committed state after an action no longer satisfies the targeted operational label; intervention precision conditions on dispatched non-abstain actions. The action-family frequencies in Table A.4 use supported randomized episodes as their denominator, with the unlisted 0.05 assigned to abstention. "Non-helpful" denotes a dispatched action that neither resolves its targeted hazard at the next committed state nor improves the terminal outcome over its paired abstention continuation.

B Paired Regime Accounting

Table A.1: Run-level win/tie/loss counts for BranchAid versus Extra-Search. Each hard row contains $3 \times 140 = 420$ pairs; easy and mid rows each contain $3 \times 80 = 240$ pairs.

Backbone	Regime	Wins	Ties	Losses
GPT-4o	Hard	22	390	8
	Easy	5	229	6
	Mid	7	226	7
Claude 3.5	Hard	21	391	8
	Easy	5	229	6
	Mid	9	225	6

Summing the mutually exclusive hard, easy, and mid rows yields the full-set Extra-Search counts in Table 4: 34/845/21 for GPT-4o and 35/845/20 for Claude 3.5.

Table A.2: Paired solve-rate differences for BranchAid versus CostRule used in Figure 2. Point differences are arithmetic contrasts of Table 5; intervals use the same issue-clustered bootstrap.

Backbone	Regime	Δ (pp)	95% CI (pp)
GPT-4o	Hard	2.6	[0.4, 4.9]
	Easy	-0.1	[-1.9, 1.6]
	Mid	-0.1	[-1.8, 1.6]
Claude 3.5	Hard	2.0	[0.2, 4.1]
	Easy	-0.2	[-1.9, 1.5]
	Mid	1.1	[-0.7, 2.8]

C Calibration, Action Value, and Ablations

The family frequencies sum to 0.95; the remainder is abstention. Direct, inverse-propensity-weighted, and doubly robust estimates agree in ranking targeted retrieval first and early narrow testing second.

The targeted removals establish that calibrated risk, supported action value, and selective execution function as one recovery mechanism.

Table A.3: Calibration and routing diagnostics for each hazard, using GPT-4o on SWE-bench Verified. Hazards are non-exclusive.

Hazard	Prevalence	ECE	Brier	Intv. precision	Resolve
Wrong-file focus	0.28	0.038	0.181	0.72	0.43
Under-tested patch	0.34	0.045	0.194	0.65	0.38
Environment mismatch	0.15	0.041	0.168	0.74	0.51
Dependency breakage	0.12	0.052	0.203	0.61	0.47
Over-wide scope	0.22	0.037	0.176	0.69	0.41
Macro average	—	0.042	0.184	0.68	0.44

Table A.4: Supported-subset intervention analysis on $n = 247$ decision episodes. Lift is terminal solve-rate change over abstention. Minimum supported propensity is 0.08; weights are clipped at 10, giving effective sample size 182.

Intervention	Frequency	Lift (pp), 95% CI	Savings (%)	Resolve	Non-helpful (%)
Targeted retrieval	0.31	2.8 [1.1, 4.5]	14.3	0.46	18
Static check	0.14	0.9 [-0.3, 2.1]	6.1	0.39	27
Early narrow test	0.23	2.1 [0.6, 3.7]	21.7	0.52	14
Patch decomposition	0.18	1.4 [0.1, 2.8]	11.2	0.41	22
Build deferral	0.09	0.6 [-0.5, 1.7]	8.9	0.33	38

Table A.5: Component ablations on SWE-bench Verified with GPT-4o.

Variant	Solve (%)	Cost ($)	Wasted CI	Δ solve
BRANCHAID (full)	31.7	3.92	8.5	—
Single-score predictor	30.4	4.18	9.4	-1.3
No value head	30.1	4.31	9.8	-1.6
No randomized value data	29.8	4.09	9.1	-1.9
Hazard heads only	29.5	4.42	10.1	-2.2
Random interventions	28.9	4.55	10.6	-2.8

D HAZARD-LABEL VALIDITY

Table A.6: Independent audit of 450 prefixes, balanced across five hazards and three confidence bins. Two annotators label each prefix before adjudication.

Hazard	Precision	Recall
Wrong-file focus	0.84	0.79
Under-tested patch	0.79	0.75
Environment mismatch	0.88	0.82
Dependency breakage	0.76	0.71
Over-wide scope	0.81	0.78
Macro	0.82	0.77

Macro precision is 0.82 (95% CI [0.78, 0.85]), macro recall is 0.77 ([0.73, 0.81]), and macro F1 is 0.79. Inter-annotator agreement is 0.73, with 0.09 of prefixes adjudicated as ambiguous. The audit establishes reliable operational state variables for routing, including the more difficult dependency and under-testing categories.

E REPOSITORY SHIFT

Table A.7: Transfer with GPT-4o. Each shifted setting contains 100 evaluation issues across three seeds and a separate 200-prefix recalibration-only pool. The interval is BRANCHAID minus Extra-Search.

Setting	ECE before	ECE after	Extra-Search	CostRule	BRANCHAID	Δ 95% CI
In-domain Python	0.042	0.042	30.2	30.5	31.7	[0.3, 2.8]
Held-out Python	0.071	0.053	28.4	29.0	30.0	[0.2, 3.1]
TypeScript	0.094	0.068	27.8	28.2	29.0	[-0.2, 2.6]
Monorepo	0.092	0.071	27.3	27.6	28.0	[-0.7, 2.1]
Mixed-language	0.103	0.078	26.8	27.2	28.0	[-0.2, 2.7]

Recalibration improves ECE in every shifted setting. Held-out Python retains a positive solve interval, while the three larger ecosystem shifts overlap zero and identify where repository-specific adaptation has the highest expected return.

F END-TO-END EFFICIENCY

Table A.8: Unconditional spend and resolved-issue yield, including failures. Yield is resolved issues per $100 of total metered spend.

Backbone	Policy	Mean spend ($)	Solves per $100
GPT-4o	Extra-Search	6.24	4.84
	CostRule	5.68	5.37
	BRANCHAID	5.36	5.91
Claude 3.5	Extra-Search	5.91	5.60
	CostRule	5.32	6.32
	BRANCHAID	5.08	6.85

On one A100 80GB GPU, BRANCHAID adds 0.36 seconds median and 0.74 seconds at p95 per decision, equal to 1.8% of end-to-end wall time. The controller-inclusive accounting therefore preserves the cost advantage: GPT-4o yield rises from 4.84 to 5.91 resolved issues per $100.

G SYSTEMS ASSURANCE MATRIX

The relevant stressors are temporal-proxy injection, budget mismatch, symptom-rule competition, trajectory difficulty, repository shift, and controller overhead. Table A.9 links each protected claim to its direct test and quantitative evidence.

Table A.9: Compact assurance account for the recovery exchange.

Protected claim	Direct test	Evidence
Prefix integrity	Proxy scrub and order perturbation; invalid-future positive control	Stable valid pipelines and a detected invalid jump; Table 6
Spend parity	Event reconciliation, residual retention, and pre-dispatch cap checks	Paired solve, cost, and CI contrasts; Equation 2, Table 4
Learned value	Identical-state, identical-menu CostRule	$+1.2$ solve points on both backbones; Tables 3–4
Selective use	Regimes fixed before policy evaluation	Resolved hard-regime gains and neutral easy/mid intervals; Table 5
Shift behavior	Repository-disjoint evaluation with separate recalibration pools	Improved ECE and held-out Python gain; Table A.7
Net efficiency	Single-device timing and failure-inclusive spend	1.8% wall-time share with higher yield; Table A.8

The checks address different failure paths rather than serving as interchangeable robustness tests: prefix perturbations test information timing, CostRule tests action selection, the regime map tests selectivity, and all-run accounting tests whether controller overhead erases the resource gain.