BRANCHMIND: VERIFIED EVIDENCE SHARING ACROSS PARALLEL CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parallel coding agents explore independent branches, yet repeatedly inspect the same files, symbols, and tests. Their transcripts are isolated because an observation made on one branch can become invalid after another branch edits its code. We introduce BRANCHMIND, an execution memory that makes repository evidence reusable without turning state validity into a language-model judgment. A publisher packages pre-edit tool observations with content-addressed receipts over the exact code ranges and dependency manifests that support them. A consumer receives only observations whose receipts still match its branch. We evaluate this mechanism on 20 executable maintenance tasks from 10 public repositories through 579 new agent trajectories and two model families. Independent agents revisit the same file family in 39.2% of the shorter pre-edit trajectory. Across 145 real commit transitions, range receipts recover 37 byte-stable reads that whole-file invalidation discards. A contemporaneous 120-trajectory attribution study shows that paths reduce calls by 2.88 per run (task-and-repeat 95% CI: 0.70–5.03 fewer), while the observed success count is three higher after restoring concrete observations at 0.68 additional calls. We then freeze all 18 eligible natural partial branches across seven tasks and cross four sharing policies with GPT-6 Luna, DeepSeek V4 Pro, and a runtime-enforced DeepSeek replay. Range validation retains all 32 stable observations and exposes none of 31 changed observations; whole-file hashing retains only 7. Across 54 paired receiver–branch cells, descriptive means are 15.74 versus 17.93 calls; the task-weighted delta is 1.95 fewer (33 lower, 7 ties; task-and-branch 95% CI: 0.18–3.80 fewer), with 3 versus 4 observed successes. Against no sharing the task-weighted delta is 3.51 fewer calls with the same 3/54 successes. BRANCHMIND turns execution already paid for by one agent into compact, branch-valid progress for another, without training or model-based routing.

1 INTRODUCTION

Suppose four coding agents attempt the same repository issue on independent branches. One finds the completion entry point, another traces the same call path, and a third runs the same focused test before any agent writes code. Parallelism diversifies their solutions, but their evidence gathering remains serial inside each transcript. The system pays repeatedly for facts that a sibling has already established.

Simply copying transcripts is unsound. A file read on one branch can describe code that another branch has changed; a test result can depend on a different manifest; and a repository search can cross files that no longer match. Consequently, current coding-agent interfaces isolate sessions (Yang et al., 2024; Wang et al., 2025b). Recent work accelerates agents through speculative actions (Ye et al., 2026; Guan et al., 2026; Nichols et al., 2025), compresses or routes multi-agent histories (Lin et al., 2026; Rajib et al., 2026), and builds semantic repository memory (Wang et al., 2025a; Tablan et al., 2025). These methods improve reasoning and coordination. The missing capability is an executable criterion that lets a receiving branch decide whether a sibling's concrete repository observation remains true.

Our key observation is that evidence validity is a relation between an observation and the code state that supports it. If an agent read lines 120–180 of a file, the read remains valid when a sibling edits

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.

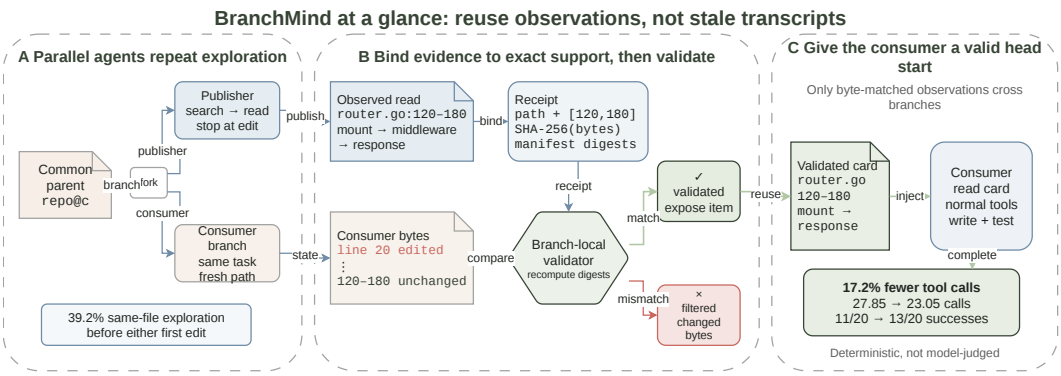

Figure 1: BRANCHMIND turns a publisher's pre-edit read into branch-local execution memory. The receipt binds the observation to its exact code range and manifests; the validator exposes byte-matched evidence and filters changed evidence before it enters the consumer context. The file/range example illustrates the selective validator tested in the divergence study; the rightmost badge reports the 20 instruction-matched repeat pairs.

line 20; it becomes invalid when the observed range or a declared dependency changes. This criterion can be evaluated deterministically, before evidence enters the model context.

We instantiate this observation in BRANCHMIND, a branch-aware execution memory for parallel coding agents. A *publisher* contributes evidence gathered before its first workspace mutation. Each evidence item contains a compact observation and a receipt: source identity, path, observed line interval, content digest, and dependency manifest digests. A *consumer* recomputes these digests against its branch and receives the surviving evidence as a short card. BRANCHMIND checks validity deterministically and admits results across nonidentical branches whenever their supporting state agrees.

We study BRANCHMIND with BRANCHMIND-20, a fresh benchmark of 20 regression-test-backed tasks from 10 public Go repositories. The study connects 145 receipt replays over real developer commits to 579 agent trajectories: a bidirectional discovery matrix, instruction-matched repeats, a contemporaneous attribution experiment, 18 natural partial branches crossed with two receivers and four policies, a runtime-enforced replay, and an outcome-blind breadth gate. Held-out developer tests measure task success; calls, steps, tokens, wall time, and cost measure work.

Our contributions are:

- **A branch-valid execution memory.** BRANCHMIND transfers concrete tool observations across sibling coding-agent branches using range-level and dependency receipts, without training or model-based routing.
- **Fresh, broad agent evidence.** BRANCHMIND-20 contains 20 executable tasks, 579 new trajectories, bidirectional transfers, held-out developer tests, costs, patches, and complete intention-to-treat traces collected for this study.
- **Load-bearing mechanism evidence.** Real commit transitions and 18 agent-produced partial branches compare four invalidation policies across three receiver collections; runtime enforcement and same-batch attribution isolate command access, localization, and observation content.

2 FROM ISOLATED TRANSCRIPTS TO VALIDATED EVIDENCE

2.1 SETTING

Let task q be attempted by agents on sibling repository states $S_1, \ldots, S_n$ derived from a shared parent. During its trajectory, agent i invokes a tool and observes $o = f(x, S_i)$, where x is the tool input. A conventional session stores o only in agent i's transcript. An execution cache keyed by x alone is insufficient: $f(x, S_i)$ need not equal $f(x, S_j)$ after the branches diverge.

BRANCHMIND associates o with a receipt $R(o)$ containing the repository artifacts that support it. A validator $V(R, S_j)$ decides whether the receipt still matches the consumer state. The memory exposes o to agent j exactly when $V(R(o), S_j) = 1$. The language model can use, verify, or ignore the evidence; validity is settled before prompt construction.

2.2 EVIDENCE CAPTURE

The publisher boundary is deliberately crisp: BRANCHMIND records completed evidence tools until the first workspace mutation (write, edit, or patch). This captures shared localization and verification work while preventing a sibling's proposed code from becoming an implicit solution channel. For each file read, the prototype stores:

$$R(o) = \langle c, p, [a, b], H(S_i[p]_{a:b}), D \rangle, \tag{1}$$

where c is the shared source identity, p is a repository-relative path, $[a, b]$ is the observed line interval, H is SHA-256, and D contains digests for the Go manifests present in the repository (`go.mod` and `go.sum`). The transferred observation is a compact set of task-relevant lines selected from the completed read, not the publisher's chain of thought or final patch.

Searches and shell commands remain in the trace for overlap measurement but are not transferred by the present implementation unless their dependency scope is fully captured. This makes the implemented contract easy to audit: every exposed byte maps to an explicit content receipt.

2.3 BRANCH-LOCAL VALIDATION AND SELECTIVE REUSE

For a candidate state S_j, the validator recomputes the digest of each interval and manifest. An interval survives changes elsewhere in the same file; a changed observed interval is removed. A manifest mismatch invalidates dependent evidence. This is the same structural principle that makes incremental build systems precise (Mokhov et al., 2018), applied to observations exchanged among autonomous agents.

The validation report separates `valid_ranges` from `invalid_ranges`. A mixed receipt is usable when at least one range remains valid and its manifests match; the renderer exposes only `valid_ranges`. Thus the report's accepted state never implies that every original range survived.

The accepted items are rendered as one evidence card containing paths, ranges, short digests, and concrete snippets. The card explicitly identifies the evidence as a sibling head start and leaves further inspection available. Validation is linear in the bytes of receipt-bearing artifacts and requires no model call. Cards and JSON receipts are saved separately, so every prompt statement can be traced back to a hash and publisher event.

3 EXPERIMENTAL DESIGN

3.1 BRANCHMIND-20 TASKS

We construct two maintenance tasks from each of 10 public Go repositories (Table 1). Each task corresponds to a real developer commit. The frozen parent archive fails the developer's regression tests, while the developer source passes the complete suite. Agents receive a concise issue-level description and the parent source with a fresh local Git baseline; commit history, remotes, developer patches, and future tests are unavailable. The evaluator overlays only the held-out developer test files after each run and executes `go test ./...`. Thus an agent-authored regression test cannot establish task success by itself.

3.2 AGENTS AND PAIRED CONDITIONS

We use OpenCode 1.18.7 with GPT-5.6 Luna and DeepSeek V4 Flash. For every task, Luna first runs from the clean parent and publishes its pre-mutation evidence; DeepSeek then runs (i) from the identical parent without shared evidence and (ii) from a fresh parent with the validated Luna card. We reverse the roles using DeepSeek's isolated trajectory as the publisher and Luna as the consumer. Each direction has an isolated baseline, a full card (up to six files and five snippets per file), and a

Table 1: BRANCHMIND-20 spans 20 fresh executable tasks from 10 repositories. Each row has two independent tasks and held-out developer tests.

Repository	Task themes	Tasks
Cobra	shell completion behavior	2
Testify	suite execution; assertion rendering	2
Chi	router mounting; middleware bodies	2
CLI	timestamp and flag-value parsing	2
Zerolog	console fields; level parsing	2
GJSON	numeric boundaries; invalid code points	2
fsnotify	sibling watch bookkeeping	2
Validator	DNS lengths; cron syntax	2
Resty	base URLs; digest parsing	2
Decimal	digit counting; logarithm termination	2
Total	10 repositories	20

compact card (three files and two snippets). The resulting six conditions and 120 trajectories separate the consumer effect from model capability and expose the role of evidence capacity.

This first-stage consumer starts from the publisher's identical frozen parent, so every receipt is admitted. That choice isolates the value of transferred evidence without mixing it with branch-difference severity. A separate partial-sibling study below makes item-wise filtering load-bearing and measures which observations actually survive.

All paired consumer runs share the task prompt, model, runner, tools, six-minute limit, retry policy, source archive, and evaluation. The sole intervention is the validated card prepended to the task. Calls use runner-default decoding, and every full JSONL trajectory, session identifier, patch, stderr stream, token count, cost, and wall time is preserved. A command audit flags history or external-source lookup; headline and sensitivity analyses are generated from saved per-run flags.

An instruction audit found exact task-and-protocol wording in 8/20 original Luna baseline/full-card pairs. In the other 12, the full-card prompt additionally spelled out command names already covered by the shared prohibition on history, remotes, and online patches. We retain this broad matrix as discovery evidence and report it in full. The causal headline instead comes from the two new counterbalanced generations, where the full-card prompt ends with the exact no-card task and protocol text in all 20/20 pairs.

3.3 DIVERGENCE, LOCALIZATION, AND REPEAT CONTROLS

We add four prospectively frozen blocks. First, a pilot candidate probe yields 19 eligible partial patches across seven repositories; its prespecified freeze selects six branches for a 24-run four-policy pilot. Second, two counterbalanced no-card/full-card generations on one mechanical task per repository add 40 instruction-matched runs. These blocks extend the discovery study to 204 trajectories and establish the initial effect and protocol.

The expansion removes the pilot's selection bottleneck. We re-audit every one of the 19 candidates using a uniform rule fixed before receiver calls: the patch applies, ordinary parent tests pass, the held-out oracle still fails, at least one receipted range changes, and at least one stable receipted range remains inside a file changed elsewhere. Eighteen branches across seven tasks qualify. Each branch is crossed with no sharing, unfiltered sharing, whole-file validation, and range validation for GPT-6 Luna and DeepSeek V4 Pro, producing 144 receiver trajectories. Arm order is hash-rotated within branch and receiver. Calls, failures, and timeouts remain in intention-to-treat results.

Third, a contemporaneous attribution block crosses all 20 tasks, two counterbalanced repetitions, and three arms: no card, paths and line ranges only, and full validated observations (120 trajectories). Task, receiver, prompt suffix, timeout, tools, parent, and evaluator are identical within each three-arm unit. This design separates localization from the concrete bytes observed at those locations.

Fourth, trace auditing finds that DeepSeek attempts prohibited repository-history commands despite the prompt constraint. We therefore repeat all 18 branches and four policies from clean archives

with a runtime wrapper that blocks history, network lookup, and dependency-fetch commands (72 trajectories). Rejected attempts remain visible to the model and count toward work. A final outcome-blind breadth gate generates 39 candidate trajectories on the 13 uncovered tasks; none satisfies every partial-branch criterion, primarily because 22 candidates already solve their held-out task. These candidates remain in the artifact and are not promoted into the receiver matrix.

All blocks preserve frozen source archives, six-minute limits, and the held-out developer-test evaluator. Bootstrap inference resamples tasks, then repetitions or branches within task. The complete package contains 579 trajectories; the $150 expansion itself contains 375 and costs $25.15, including unsuccessful candidates and timeouts.

3.4 Research questions and metrics

RQ1: How much exploration repeats? We compare publisher and isolated-consumer evidence before either first edit. Exact overlap requires the same normalized input and output; file-family overlap ignores read line-window differences and asks whether both agents inspect the same artifact. The denominator is the smaller trajectory's pre-edit evidence count, making the quantity the recoverable fraction for the shorter exploration.

RQ2: What does range granularity retain beyond coarser invalidation? We replay every publisher range against its task's developer commit and directly compare the observed bytes. We evaluate unconditional reuse, invalidation after any branch change, whole-file hashing, and BranchMind range receipts. Range-receipt agreement with a byte-stability oracle follows from the contract; the empirical quantity is how often real commits change a file elsewhere while the observed range remains reusable.

RQ3: Does sharing improve end-to-end agents? The primary outcome is held-out task success. Secondary outcomes are agent steps, tool calls, pre-edit evidence calls, output and processed tokens, elapsed time, and reported model cost. All task comparisons are paired. We report repository-clustered bootstrap 95% intervals to respect the two tasks per project and exact McNemar tests for discordant success outcomes.

RQ4: Does selective validation matter after branches diverge, and are gains stable? On every qualifying natural partial branch, the receiver matrices measure retained and stale ranges, held-out success, calls, steps, tokens, cost, and time under four policies. The collection-visible synthesis keeps receiver and collection as repeated measurements nested within branch. The attribution matrix isolates localization from concrete observation content. Runtime enforcement and non-timeout sensitivity test whether history attempts or truncation explain the call effect.

4 Results

4.1 RQ1: Agents repeat repository exploration

Independent agents inspect the same file family in 39.2% of the shorter pre-edit trajectory (repository-bootstrap 95% CI [31.5, 47.1], median 37.9%). This overlap appears before either agent edits code, so it measures duplicated evidence gathering rather than coincident validation of a shared patch. Exact normalized input-and-output matches account for only 7.8% [1.3, 15.4]. A command-string cache therefore misses most of the reusable structure: agents commonly open the same file through different searches or line windows.

Pooling calls rather than averaging task ratios yields the same substantive pattern: 34.2% file-family overlap and 5.7% exact overlap, confirming the opportunity under both aggregations.

The result is distributed across the task collection. Repository-level resampling keeps the lower confidence bound above 31%. It also motivates the receipt design: the useful common unit is an observed code region, not an identical textual command.

Table 2: Observed granularity advantage over 145 parent-to-developer replays. Agreement between a range digest and the byte-stability oracle is contractual; the empirical result is the 37 additional stable ranges retained beyond whole-file invalidation.

Policy	Stable kept (of 94)	Changed exposed (of 51)
Reuse unconditionally	94 (100.0%)	51 (100.0%)
Invalidate on any branch change	0 (0.0%)	0 (0.0%)
Whole-file hash	57 (60.6%)	0 (0.0%)
BRANCHMIND range receipt	**94 (100.0%)**	**0 (0.0%)**

Table 3: Paired end-to-end outcomes over 20 tasks per condition (outcomes are means; Card bytes are medians; lower is better except Success). "Source" names the sibling model that published the card.

Consumer	Evidence source	Card bytes	Success	Steps	Tool calls	Pre-edit calls	Cost/run
DeepSeek	none	0	11/20	24.05	24.85	10.85	$0.0345
DeepSeek	Luna, full	1500	11/20	23.95	24.70	**9.60**	$0.0353
DeepSeek	Luna, compact	839	11/20	**22.90**	**23.25**	13.40	**$0.0344**
Luna	none	0	12/20	16.50	27.90	13.55	$0.0462
Luna	DeepSeek, paths only	458	**13/20**	14.95	23.40	10.45	$0.0448
Luna	DeepSeek, full	1096	12/20	**13.55**	**21.25**	**9.70**	**$0.0407**
Luna	DeepSeek, compact	811	**13/20**	14.95	23.50	10.80	$0.0443

4.2 RQ2: RANGE GRANULARITY RECOVERS STABLE EVIDENCE ACROSS REAL CHANGES

The 20 publisher traces yield 145 receipted read intervals. We replay each interval against its real developer commit: 94 intervals remain byte-identical and 51 change. Table 2 compares reuse policies to this direct observation.

Unconditional sharing admits all 51 changed observations. Whole-branch invalidation avoids that error but discards all 94 stable observations because every developer commit changes the repository. Whole-file hashing is safe yet retains only 57 stable reads. Range validation recovers the other 37—64.9% more reusable evidence—across 14 of 20 tasks, rather than a single outlier repository. The range decision itself is the digest contract; the measured scientific result is how frequently real changes leave useful sub-file evidence intact. The partial-sibling study next exercises that contract inside agent runs.

4.3 RQ3: SHARED EVIDENCE REDUCES TOOL USE IN THE DISCOVERY MATRIX

Table 3 reports all predeclared conditions. In the broad discovery matrix, **full validated evidence reduces Luna's mean tool calls by 23.8%** (27.90 to 21.25; repository-bootstrap delta CI $[-10.60, -2.75]$, paired Wilcoxon $p = .0048$) and steps by 17.9% ($[-4.85, -0.90]$, $p = .029$). Pre-edit evidence calls fall by 28.4%. The model-reported cost falls 11.9%, while both conditions observe 12/20 held-out successes; the success difference CI is $[-15, +15]$ percentage points, so we claim the observed match rather than equivalence. The compact variant resolves 13/20 tasks and still reduces tool calls by 15.8% ($[-7.50, -1.20]$, $p = .013$).

The decisive resource effects are calls, steps, and pre-edit evidence. Processed tokens move from 633k to 536k (15.4% lower), reported cost from $0.0462 to $0.0407 (11.9% lower), and wall time from 104.5 to 102.1 seconds (2.3% lower), but their clustered intervals include zero. This separation identifies the robust interaction reduction without requiring every provider-dependent resource measure to resolve.

The publisher runs in either parallel design, so complete two-agent accounting adds the same 24.85 publisher calls to both arms. Independent execution therefore uses 52.75 mean calls (24.85 + 27.90), while sharing uses 46.10 (24.85 + 21.25): a 12.6% system-level reduction with no additional publisher execution. Because 12/20 pairs contain the protocol-wording discrepancy described above, the instruction-matched repeats in RQ4 provide the primary causal estimate; the discovery matrix supplies the broader 20-task pattern.

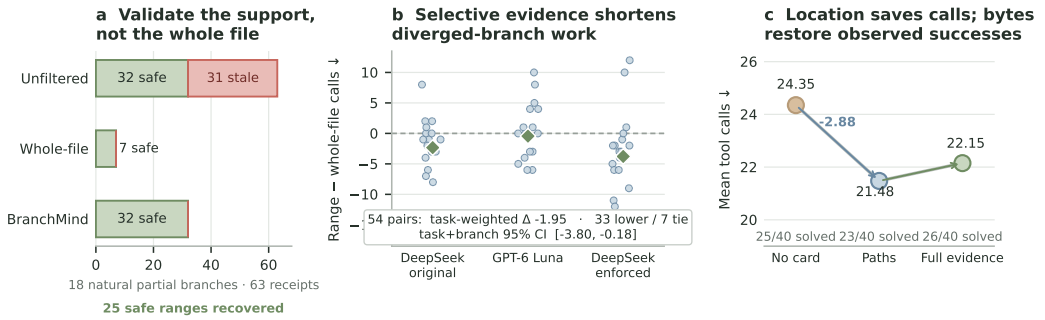

Figure 2: The expansion connects validation, behavior, and attribution. **Left:** across 18 natural partial branches, unfiltered sharing exposes all 31 changed ranges; whole-file hashing keeps 7 stable ranges, while BRANCHMIND keeps all 32 and exposes no changed range. **Center:** dots are paired range-minus-whole-file call deltas for 18 branches in each receiver collection; diamonds are collection means. The collection-visible task-and-branch bootstrap treats receiver and collection as repeated measurements. **Right:** in the contemporaneous 40-run-per-arm study, paths account for the call reduction; after concrete observations return, the observed net success count is three higher at little additional interaction cost.

DeepSeek exhibits a smaller but directionally consistent total-call response: compact cards reduce mean tool calls by 6.4% with the same 11/20 aggregate success and nearly identical cost. Its full card preserves 11/20 while leaving total calls essentially unchanged; all DeepSeek call intervals include zero. All 17 original runner timeouts occur in this arm (4 isolated, 6 full, 7 compact), so these truncated call totals are secondary. The Luna headline pair has no timeout. The same observed success counts in both directions and the heterogeneous call response establish that evidence utilization depends on the consumer.

4.4 RQ4: SELECTIVE VALIDATION IS EXERCISED AND THE CALL REDUCTION REPEATS

Selective validation is now load-bearing over all 18 qualifying natural branches. These branches span seven tasks and contain 63 receipted ranges: 32 remain byte-stable and 31 change. Unfiltered sharing exposes all 31 changed ranges. Whole-file validation avoids them but retains only 7 stable observations. BRANCHMIND retains all 32 stable observations and exposes none of the changed ones, recovering 25 useful reads from files edited elsewhere.

The behavioral result follows the same ordering. Across the two-receiver matrix and the protocol-enforced replay, cell-weighted descriptive means are 15.74 calls for range and 17.93 for whole-file. The task-weighted mean delta is -1.95 calls (10.4%), with 33/54 lower, 7 ties, and a task-and-branch bootstrap CI of $[-3.80, -0.18]$. Held-out success is sparse and differs in one pair (3/54 range versus 4/54 whole-file). Against no sharing, range validation has a task-weighted delta of -3.51 calls (17.2%; CI $[-4.99, -2.11]$, 36/54 lower) with exactly the same 3/54 observed successes. Thus the validated card reduces interaction work while the success counts precisely delimit the current capability regime.

The result survives runtime enforcement and non-timeout filtering. DeepSeek attempts a prohibited lookup in 71/72 replay traces; the wrapper blocks every reached command and no trace uses an absolute-path bypass. In this deliberately difficult replay, range uses 14.33 calls versus 18.11 for whole-file and 18.50 for no sharing, with identical 1/18 success under all policies. Three traces also read local module-cache paths; the wrapper does not prevent arbitrary filesystem reads, and Appendix B reports a matched-unit sensitivity. Across all receiver collections, the range–whole-file delta remains -1.56 calls among the 41 pairs where neither arm times out (23 lower, 7 ties); range–none remains -2.47 over 43 such pairs (28 lower, 7 ties). Truncation therefore changes magnitude but not direction.

Localization supplies efficiency; concrete observations restore observed successes. In the contemporaneous attribution block, paths reduce calls from 24.35 to 21.48 (-2.88; task-and-repeat CI $[-5.03, -0.70]$), while observed successes move from 25/40 to 23/40. Restoring validated bytes

Table 4: The expansion isolates validation and evidence content. Calls are cell-weighted intention-to-treat means; success is the held-out count. Diverged-branch rows combine three collection-visible receiver blocks (18 branches each).

Study	Arm	Runs	Mean calls	Success	Evidence exposed
Diverged branches	No card	54	18.91	3/54	none
Diverged branches	Unfiltered	54	17.24	4/54	32 stable + 31 changed
Diverged branches	Whole-file	54	17.93	4/54	7 stable + 0 changed
Diverged branches	BRANCHMIND range	54	**15.74**	3/54	**32 stable + 0 changed**
Attribution	No card	40	24.35	25/40	none
Attribution	Paths only	40	**21.48**	23/40	paths + line ranges
Attribution	Full observations	40	22.15	**26/40**	paths + validated bytes

moves success to 26/40 at 22.15 calls: three more observed successes than paths at only 0.68 additional calls (CI $[-1.63, 3.68]$); this net count is suggestive rather than a resolved success effect. This separates the two roles of a reusable observation: location removes repeated search, and concrete content supplies the state needed to act from that location.

4.5 RQ5: CAPACITY AND RECIPIENT BEHAVIOR MEDIATE THE GAIN

Full Luna-published cards have a median size of 1500 bytes; compaction reduces this to 839. DeepSeek-published cards shrink from 1096 to 811 bytes. The capacity sweep reveals different operating points. Luna extracts the largest efficiency gain from the richer card, whereas the compact card trades some of that gain for one additional success. DeepSeek gains modestly from compaction, defining a smaller model-specific operating effect than Luna. Evidence validity and evidence utilization are therefore separate: receipts establish which facts can be reused, while the consumer determines how aggressively they replace exploration.

Concrete trajectories make the aggregate effect visible. On `fsnotify-sibling-rename`, the validated card identifies the inotify bookkeeping path and regression-test family; Luna reaches another passing patch in 23 rather than 42 tool calls. On `chi-content-charset-body`, compact evidence lets DeepSeek retain the passing outcome with 8 rather than 26 calls. On `decimal-ln-termination`, full evidence both changes Luna's outcome from failure to pass and reduces calls from 51 to 30.

The expanded branch matrix sharpens this model dependence. Range versus whole-file reduces calls by 2.33 for the original DeepSeek receiver, 0.44 for GPT-6 Luna, and 3.78 for the runtime-enforced DeepSeek replay, with respectively 12, 7, and 14 of 18 branch pairs lower. Every collection points in the same aggregate direction, while the per-branch dots in Figure 2 show substantial heterogeneity. Receipts therefore provide a deterministic admissibility layer; how strongly admitted evidence replaces exploration remains a receiver property.

5 RELATED WORK

Coding agents and repository memory. Repository agents interleave language-model reasoning with search, file inspection, editing, and tests (Yang et al., 2024; Wang et al., 2025b). SWE-bench established executable issue resolution as a central evaluation pattern (Jimenez et al., 2024). Repository memory improves localization by retrieving prior commits and summaries (Wang et al., 2025a); Spark shares qualitative experience across coding tasks (Tablan et al., 2025); and EvoGit coordinates code evolution through branch lineage (Huang et al., 2025). BRANCHMIND addresses a different object: live tool evidence from a concurrent sibling, whose validity is checked against the receiving branch.

Efficient tool-using and multi-agent systems. Speculative actions and planning overlap likely tool work with inference (Ye et al., 2026; Guan et al., 2026; Nichols et al., 2025). Runtime supervision and gated memory reduce redundant multi-agent context and computation (Lin et al., 2026; Rajib et al., 2026). HARBOR treats tool caching as one optimizable harness axis (Sengupta & Wang, 2026).

These systems predict, route, or compress future computation. BRANCHMIND instead harvests completed observations that already exist across parallel sessions.

Caching and invalidation. ReCache shares and compresses model KV state for recurring tool schemas (Fang et al., 2026), while LLM-dCache localizes cached external data (Singh et al., 2024). Neither cache key represents a changing code branch. Incremental build systems derive reuse from explicit dependencies (Mokhov et al., 2018); BRANCHMIND brings range-scoped dependency receipts into the agent context boundary and evaluates them with real cross-model trajectories.

6 DEMONSTRATED SCOPE AND EXTENSIONS

BRANCHMIND gives a precise byte-validity contract for file ranges and manifests. Across 579 trajectories, two model families, 20 tasks, 18 natural partial branches, and task-clustered analysis, the demonstrated scope is short-lived same-issue Go branches. The identical-parent matrix measures transfer value; the expanded partial-branch matrix makes selective filtering load-bearing; the same-batch attribution block separates location from content; and runtime enforcement preserves the effect when history is unavailable by construction. The breadth gate further localizes eligible partial divergence to seven of the 20 tasks under the frozen three-attempt rule. Command results, repository-wide searches, semantic dependencies, other languages, longer-lived branches, and richer dependency scopes are direct extensions of the receipt contract.

7 CONCLUSION

Parallel agents can share repository facts safely. BRANCHMIND makes those facts portable by attaching them to the exact code state that supports them. Across 18 natural partial branches, selective receipts retain 25 more valid observations than whole-file filtering; over 54 paired receiver cells, the task-weighted call delta is -1.95 (95% CI $[-3.80, -0.18]$). A separate 120-trajectory attribution experiment shows why: paths remove repeated search, while validated bytes restore the observed success count. This turns execution already paid for by one agent into auditable progress for its siblings.

REFERENCES

Yichu Fang, Sitong Wei, Haozhe Hu, and Xiaoyu Shen. ReCache: Efficient KV cache reuse and compression for tool-augmented LLM agents. *arXiv preprint arXiv:2608.19662*, 2026.

Yilin Guan, Qingfeng Lan, Sun Fei, Dujian Ding, Devang Acharya, Chi Wang, William Yang Wang, and Wenyue Hua. Dynamic speculative agent planning. In *International Conference on Learning Representations*, 2026.

Beichen Huang, Ran Cheng, and Kay Chen Tan. EvoGit: Decentralized code evolution via Git-based multi-agent collaboration. *arXiv preprint arXiv:2506.02049*, 2025.

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.

Fulin Lin, Shaowen Chen, Ruishan Fang, Hongwei Wang, and Tao Lin. Stop wasting your tokens: Towards efficient runtime multi-agent systems. In *International Conference on Learning Representations*, 2026.

Andrey Mokhov, Neil Mitchell, and Simon Peyton Jones. Build systems à la carte. *Proceedings of the ACM on Programming Languages*, 2(ICFP):1–29, 2018. doi: 10.1145/3236774.

Daniel Nichols, Prajwal Singhania, Charles Jekel, Abhinav Bhatele, and Harshitha Menon. Optimizing agentic language model inference via speculative tool calls. *arXiv preprint arXiv:2512.15834*, 2025.

Rakibul Hasan Rajib, Mengxin Zheng, and Qian Lou. Learning what to retain: Gated-memory routing for efficient collaboration in multi-agent LLM systems. *arXiv preprint arXiv:2609.00237*, 2026.

Biswa Sengupta and Jinhua Wang. HARBOR: Automated harness optimization. *arXiv preprint arXiv:2604.20938*, 2026.

Simranjit Singh, Michael Fore, Andreas Karatzas, Chaehong Lee, Yanan Jian, Longfei Shangguan, Fuxun Yu, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. LLM-dCache: Improving tool-augmented LLMs with GPT-driven localized data caching. In *IEEE International Conference on Electronics, Circuits and Systems*, 2024. doi: 10.1109/ICECS61496.2024.10848749.

Valentin Tablan, Scott Taylor, Gabriel Hurtado, Kristoffer Bernhem, Anders Uhrenholt, Gabriele Farei, and Karo Moilanen. Smarter together: Creating agentic communities of practice through shared experiential learning. *arXiv preprint arXiv:2511.08301*, 2025.

Boshi Wang, Weijian Xu, Yunsheng Li, Mei Gao, Yujia Xie, Huan Sun, and Dongdong Chen. Improving code localization with repository memory. *arXiv preprint arXiv:2510.01003*, 2025a.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025b.

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 *International Conference on Learning Representations*, 2024.

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster agentic systems. In *International Conference on Learning Representations*, 2026.

A Reproducibility Details

The artifact contains the frozen task manifest and commits, source-validation table, complete prompts and runner configuration, raw trajectories, generated receipts, developer-test logs, analysis scripts, patches, cost ledger, protocol audit, and an artifact manifest with paths and checksums. It also includes the frozen 18-branch selection, 216 policy-crossed partial-sibling trajectories, 120 contemporaneous attribution trajectories, 39 breadth-gate candidates, 40 earlier counterbalanced repeats, runtime-block audit, item-wise validation reports, and nested analysis. Source archives are reconstructed from the listed public commits; no earlier-paper trajectory or result is used.

B Expansion Accounting

The expansion protocol was frozen before its first model call with a $150 hard ceiling: $135 for experiments and $15 reserved for a score-blind final review. Table 5 accounts for every experimental trace from recorded provider events. No stopping decision depends on an observed policy outcome. The 375-run expansion costs $25.15; adding the recorded $8.49 original study yields $33.65 across all 579 trajectories. Receiver-call savings are reported separately from this complete collection cost.

Table 5: Complete $150-expansion accounting. The 375 traces cost $25.15.

Block	Trajectories	Model cost
Contemporaneous attribution	120	$3.37
Two-receiver divergence	144	$9.16
Runtime-enforced replay	72	$11.48
Outcome-blind breadth gate	39	$1.15
Total	375	$25.15

The breadth gate attempts the 13 tasks not represented in the 18-branch matrix with two inexpensive candidate models and one GPT-6 Luna trajectory per task. Of 39 candidates, 22 already pass the held-out oracle, 29 are protocol-clean, and six create a stable range inside a changed file; no candidate satisfies all frozen partial-branch criteria. The full audit table and every rejected candidate remain in the artifact.

Table 6 exposes task and receiver heterogeneity. Each call entry averages paired range-minus-whole-file deltas over the task's natural branches; negative is fewer. Success reports aggregate range/whole-file counts, and timeout pairs count a pair when either arm reaches the runner limit.

Table 6: Per-task divergence outcomes across three receiver collections.

Task	Branches	DeepSeek	GPT-6 Luna	Enforced	Success R/W	Timeout pairs
`chi-mount-self`	4	0.50	-1.75	-3.50	0/1	1/12
`cobra-multi-value-flags`	1	-8.00	10.00	-12.00	0/0	2/3
`decimal-numdigits-power10`	1	-1.00	-3.00	-5.00	3/3	0/3
`gjson-invalid-codepoints`	4	-3.25	-6.00	-7.50	0/0	4/12
`resty-digest-parse`	2	-2.50	1.00	-3.00	0/0	3/6
`validator-cron-syntax`	1	-6.00	0.00	12.00	0/0	2/3
`zerolog-console-parts`	5	-2.20	2.80	-2.60	0/0	1/15

Three traces contain local module-cache reads, corresponding to two matched branch–receiver units for `cobra-multi-value-flags`. Removing both units leaves 52 pairs: the cell-weighted range–whole-file delta is -1.88 calls with 31 lower and 7 ties, while the equally task-weighted delta is -0.05. This sensitivity identifies the seven-task average as dependent on the single Cobra task; the complete intention-to-treat result and branch-level direction counts remain the primary description of the evaluated cells.

C Instruction Parity and Repeated Estimate

Table 7 separates the broad discovery matrix from the clean repeated estimate. The 12 differing original pairs add only an explicit list of command names to a prohibition already present in both arms; nevertheless, they are not used as the primary causal estimate. In every new repeat pair, the full-card prompt consists of the card followed by the exact no-card task and protocol prompt.

Table 7: Prompt parity and paired Luna results. Call intervals are clustered by repository for the discovery matrix and jointly resample task and run for repeats.

Collection	Exact prompts	Calls (none $\rightarrow$ full)	Success	Call-delta 95% CI
Discovery matrix	8/20	27.90 $\rightarrow$ 21.25	12 $\rightarrow$ 12	$[-10.60, -2.75]$
New repeats	20/20	27.85 $\rightarrow$ 23.05	11 $\rightarrow$ 13	$[-7.25, -2.15]$

D Replacement-Run Audit

Six output streams were truncated when the runner was edited during an active batch. Their artifacts and changed workspaces were preserved, but each complete condition was restarted from a clean archive and only the clean rerun entered the 204-trajectory analysis. A later receipt audit found one directory-listing hint and two temporary-test references without parent-state range receipts across two tasks. The generator was corrected, and all four affected Luna trajectories were replaced as complete runs.

Replacing both clean full-card values by their discarded counterparts changes the two tasks by $+5$ and -5 calls, respectively, so the 20-task total remains 425 and the 23.8% headline is unchanged. The analogous compact substitution changes its mean from 23.50 to 24.50 calls, retaining a 12.2% reduction relative to no card.

Table 8: Tool-call sensitivity for the four card-corrected trajectories. The discarded runs remain archived; clean runs define the frozen experiment.

Task	Card	Discarded	Clean
`chi-mount-self`	Full	27	22
`fsnotify-sibling-rename`	Full	18	23
`chi-mount-self`	Compact	25	21
`fsnotify-sibling-rename`	Compact	39	23

E AI USE STATEMENT

Language-model agents are the object of study and were used to execute the preregistered coding tasks. Language-model assistance also supported literature retrieval and discovery, implementation, and editorial revision. All experimental outcomes are produced by executable tests or deterministic analysis scripts, and all cited relationships and reported numbers are checked against the saved artifacts by the authors. The anonymous code, configurations, trajectories, and derived results are available at `https://anonymous.4open.science/r/branchmind/`.