The Handoff Tax: Continuing Non-Native Trajectories in LLM Agents

Roy Ganz [*] [†], **Mor Shpigel Nacson**[*], **Adi Kalyanpur, Ron Litman**

AWS, Agentic AI

Abstract

Coding agents perform long-running tasks spanning dozens of model calls, tool uses, and code edits. As these runs unfold, users face a practical cost–quality trade-off: *escalating* to a stronger model when a cheaper one struggles, or *downshifting* once the hard reasoning is complete. Each switch requires the receiver to continue a *non-native* trajectory produced by another model. We study how this handoff affects quality and cost, and how varying the trajectory information inherited by the receiver changes the outcome. Using pairs of low-cost, low-capability (LC) and high-cost, high-capability (HC) models from the Claude and GPT families, we vary handoff direction, timing, and interface, comparing full-trajectory transfer, compaction, and trajectory removal while preserving the repository state. Across both model families, full-trajectory escalation recovers less than half of the LC-to-HC quality gap while incurring a substantial cost premium. We term this cost–quality penalty the *handoff tax*. By contrast, downshift offers a favorable cost–quality point. Interestingly, the preferred interface also reverses with direction: reducing LC-model trajectory information improves escalation quality, whereas removing the HC-model trajectory reduces downshift quality.

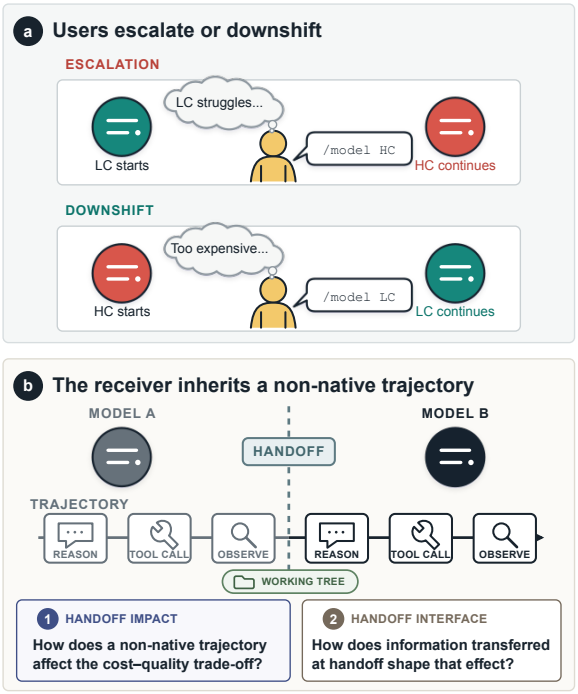

Figure 1: **The Handoff Tax. (a)** Users escalate to a high cost model (HC) when the cheaper model (LC) struggles, or downshift to a cheaper one once hard reasoning appears complete. **(b)** At handoff, the receiver inherits another model's trajectory. We ask how this affects cost and quality, and how varying the transferred trajectory information shapes the effect.

1 Introduction

Coding agents now routinely chain dozens to hundreds of model calls on a single task, making model choice an economic decision as much as a technical one (Yao et al., 2023; Schick et al., 2023; Qin et al., 2023; Patil et al., 2023; Wang et al., 2024, 2025; Xu et al., 2025; Hitzig et al., 2026). Available models span distinct cost–quality profiles: higher-capability models resolve more issues but are substantially more expensive, whereas cheaper models offer lower cost at reduced capability. Users must often choose among these models before knowing what level of capability the task actually requires. Model switching offers a natural response and is already supported directly by coding-agent products. Users may start with a low-capability model and *escalate* to a stronger one when the agent struggles; conversely, once the hard reasoning appears complete, they may *downshift* to a cheaper model to finish the task at lower cost. Throughout, **LC** and **HC** denote the lower-cost/lower-capability and higher-cost/higher-capability models within a family; in our experiments, these correspond to Claude and GPT model pairs. Despite being exposed in

[*] Equal contribution.
[†] Correspondence: royganz@amazon.com.

several coding-agent interfaces (e.g., `/model` commands in Kiro, Codex, and Claude Code), the cost–quality consequences of capability handoffs in long-horizon coding agents remain poorly understood.

A mid-task handoff places a model in an unusual position: it must continue a long trajectory that *another* model produced. Unlike ordinary generation, where models extend their own trajectory–their own phrasing, hypotheses, tool-use idioms, and dead ends–a handoff requires them to inherit a trajectory they did not create. This inherited context may contain reasoning the receiver would not have generated and mistakes it would not have made. Whether it helps or hurts may depend on direction: an HC model might be *anchored* by LC's wrong turns, while an LC model might coast on HC's groundwork–or flounder when extending reasoning beyond its own ability.

We therefore ask two questions: whether and how inheriting a non-native trajectory changes the cost–quality trade-off, and how the information transferred at the handoff shapes that effect. We study these questions systematically on SWE-bench Verified (Jimenez et al., 2024) along three axes: the *direction* of the switch (LC→HC and HC→LC), its *timing* (swept across difficulty-calibrated percentiles), and the *interface*, which determines what trajectory information the receiver inherits. The standard *raw handoff* passes the full trajectory verbatim. Alternatively, the trajectory can be compacted into a summary written by either the outgoing or incoming model, or dropped while preserving the code edits already written to disk (*Traj-drop*). Fig. 2 illustrates these interfaces.

Findings and contributions. We present, to our knowledge, the first systematic study of mid-trajectory model capability handoffs in long-horizon coding agents. Across two model families, both handoff directions, seven switch points, and four interfaces, our primary study comprises **58 configurations per family, 58,000 runs, and 36 billion tokens processed** overall. Three main findings emerge from the coding-agent study:

1. Raw escalation recovers limited quality and can be dominated by restarting with HC. Raw continuation, the default escalation interface, recovers less than half of HC's quality advantage in both model families. Most strikingly, for Claude, even after paying for the LC prefix, abandoning the attempt and restarting with HC is cheaper and more accurate than Raw continuation.

2. Continuing another model's trajectory has direction-dependent value. Unlike escalation, Raw downshift provides a favorable cost–quality trade-off: LC receivers retain substantial quality gains over LC while remaining cheaper than HC. The balance differs across families: Claude retains most of LC's cost advantage, whereas GPT retains most of HC's quality advantage.

3. Handoff interfaces reveal what trajectory information helps the receiver. In escalation, reducing LC trajectory information improves the trade-off: compaction favors savings, whereas trajectory removal favors quality recovery. In downshift, removing the HC trajectory while preserving its working-tree edits reduces quality across both model families. This reveals a directional duality: HC trajectories guide LC receivers, whereas LC trajectories burden HC receivers.

2 Related Work

Prior work improves the cost–quality trade-off of LLM inference through model selection. Cascades invoke higher-capability models when lower-cost ones are insufficient, while routers dispatch requests over a model pool (Chen et al., 2023; Yue et al., 2024; Hu et al., 2024; Ding et al., 2024; Aggarwal et al., 2025; Valkanas et al., 2025; Ong et al., 2025; Kotte, 2026). Recent work extends routing to multi-turn interactions, deciding which model acts and when (Zhang et al., 2026a,b; Hemadri et al., 2026). SWE-Router (Son et al., 2026) uses an LC model's partial coding trajectory to decide whether it should continue or an HC model should restart, but does not transfer the trajectory. Khraishi et al. (2026) study directional drift when one model continues another's dialogue prefix, but only at the final turn and without alternative handoff interfaces. KC and Budathoki (2026) study agents taking over interrupted coding tasks, examining how the information exposed from the prior run affects the effort needed to rediscover the predecessor's work. Their models are evaluated as successors rather than as LC–HC alternatives with distinct cost profiles.

Our work instead studies a practical user pattern: switching during an active run between commercial models with different capability and cost profiles. We vary direction, timing, and interface to measure how continuing a *non-native* trajectory affects end-to-end quality and monetary cost.

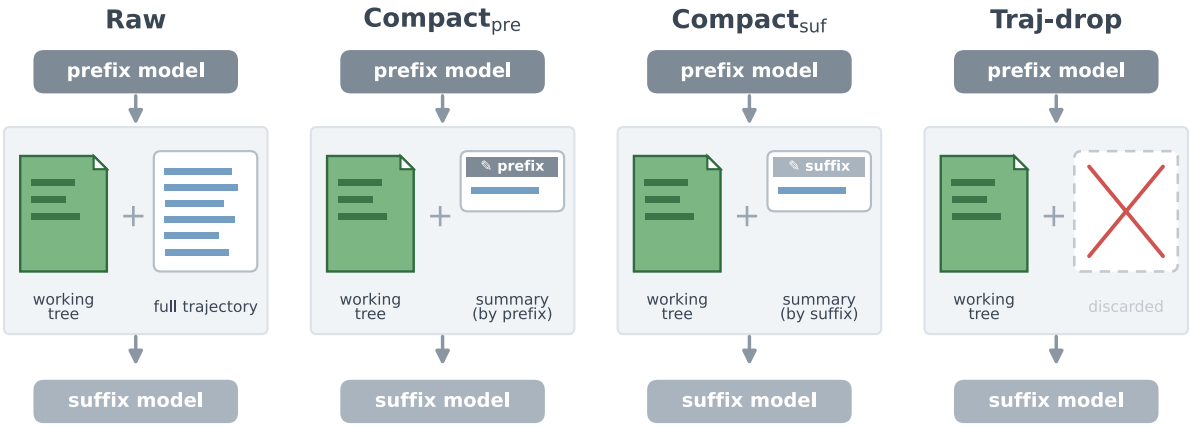

Figure 2: **Handoff interfaces.** All strategies preserve the prefix model's edited working tree at the switch but vary the trajectory information transferred to the suffix: the full trajectory (**Raw**), a summary written by the prefix or suffix model (**Compact**$_{\text{pre}}$/**Compact**$_{\text{suf}}$), or none (**Traj-drop**).

3 Experimental Framework

We study *mid-trajectory model handoffs* in long-horizon coding tasks, using SWE-bench Verified (Jimenez et al., 2024), a real GitHub issue-resolution benchmark with executable pass/fail evaluation and difficulty annotations. All conditions use the same `mini-swe-agent`[1] scaffold, tools, and prompts (Yang et al., 2024). Within each family, **LC** denotes the lower-cost, lower-capability model and **HC** the higher-cost, higher-capability model: Haiku 4.5 (Anthropic, 2025) and Opus 4.7 (Anthropic, 2026) for Claude, and GPT-5.6 Luna and Sol for GPT (OpenAI, 2026).

3.1 Handoff Conditions

Directions. We evaluate the resulting cost–quality trade-off in both model switch directions. *Escalation* (LC→HC) starts with LC and switches to HC. It tests whether HC can rescue a trajectory begun by LC and in what cost. *Downshift* (HC→LC) starts with HC and transfers the remaining work to LC. It tests how much of HC's quality advantage remains and how it affects the cost.

Handoff Strategies. The central design variable is *what crosses the model boundary*. Let $\mathcal{T}_{1:K}$ denote the trajectory produced by the prefix model up to step K, and $\mathcal{W}_K$ the working tree (edited files on disk) at that point. All strategies preserve $\mathcal{W}_K$; they differ only in the trajectory information passed to the suffix model (Figure 2):

- *Raw*: the full trajectory $\mathcal{T}_{1:K}$ is transferred.
- *Compact*$_{\text{pre}}$: the prefix model compacts $\mathcal{T}_{1:K}$; only the summary is passed.
- *Compact*$_{\text{suf}}$: the suffix model compacts $\mathcal{T}_{1:K}$ and continues from that summary alone.
- *Traj-drop*: no trajectory information is transferred; the suffix model begins with only $\mathcal{W}_K$.

For escalation, we construct two restart controls that charge for LC's work but discard both its trajectory and working-tree edits before restarting HC from the original task state. *Abort + HC fresh* charges for LC through step K, whereas *LC-full + HC-full* charges for a complete LC run. These test whether *any* continuation of the LC trajectory is preferable to abandoning it outright.

3.2 Switch Points

A handoff's effect depends on *when* it occurs. Fixed step counts are not comparable across tasks or models because trajectory lengths vary. We therefore define switch points as percentiles of the starting model's step-count distribution. Computing these percentiles over the full dataset would skew the difficulty distribution at each switch point relative to the full dataset, since easier tasks tend to finish earlier than harder ones. To avoid this bias, we estimate each percentile separately within each task-difficulty bucket. For example, p25 is the 25th-percentile step count for the relevant starting model and difficulty bucket. We sweep K over percentiles $\{5, 10, 15, 25, 35, 45, 50\}$. The main results average over all seven switch points. We omit later percentiles because too few instances reach the switch for reliable comparison. Appendix Tab. 11 report the exact bucket-specific steps.

[1] https://github.com/swe-agent/mini-swe-agent

3.3 Evaluation on the Switched Subset

Comparing handoffs fairly requires accounting for which runs actually underwent a handoff. If the starting model finishes before step K, no switch occurs; the run is a single-model trajectory and provides no evidence about the handoff effect. We therefore evaluate the **switched subset**: for each direction and K, we restrict comparisons to the intersection of instances that switched under all four strategies. This ensures that every strategy is evaluated on the same tasks that underwent a handoff. We evaluate the baselines on this same subset to provide matched quality and cost references.

3.4 Metrics

We report raw pass rate, mean cost (USD), and mean step count. Costs follow provider pricing, including cache reads and writes. We compute these metrics separately at each switch point and average them within each reported window.

Normalized metrics. Following Sec. 3.3, we evaluate each strategy at each switch point relative to the LC-only and HC-only baselines on its matched switched subset, placing all switch points on the same LC-to-HC reference scale. Let $R_m(K)$ and $C_m(K)$ denote pass rate and mean cost for strategy m; LC/HC subscripts denote the matched single-model baselines. **Quality Recovery (QRec)** is the fraction of HC's quality advantage over LC that the strategy recovers, while **Cost-Savings Retention (CSRet)** is the fraction of LC's cost advantage over HC that it retains:

$$\mathrm{QRec}(m,K) = 100\frac{R_m(K) - R_{\mathrm{LC}}(K)}{R_{\mathrm{HC}}(K) - R_{\mathrm{LC}}(K)}, \quad (1)$$

$$\mathrm{CSRet}(m,K) = 100\frac{C_{\mathrm{HC}}(K) - C_m(K)}{C_{\mathrm{HC}}(K) - C_{\mathrm{LC}}(K)}. \quad (2)$$

QRec is 0 at LC quality and 100 at HC quality; CSRet is 100 at LC cost and 0 at HC cost. Values outside $[0, 100]$ fall beyond these anchors (*e.g.*, negative CSRet is costlier than HC-only).

Experimental scale. For each model family, we evaluate 58 configurations: two single-model baselines and 56 handoff configurations spanning seven switch points, four handoff strategies, and two directions. Running all configurations on the 500 SWE-bench Verified instances across both model families yields 58,000 agent runs, 2 million LLM API calls, and 36 billion processed tokens.

4 The Handoff Tax

We evaluate both switching directions under the protocol of Sec. 3 and report matched switched-subset results in Tab. 1. Across both model families, two qualitative patterns emerge. First, Raw handoff recovers less than half of HC's quality advantage in escalation, but offers a favorable cost–quality trade-off in downshift at lower cost than HC-only. Second, inherited trajectory has direction-dependent value: dropping LC trajectory improves escalation, whereas preserving HC trajectory improves downshift. These quality patterns recur across families, while their cost implications differ because the model pairs have different baseline trajectory lengths and HC-to-LC per-task cost ratios. We examine escalation (Sec. 4.1), downshift (Sec. 4.2), and the receiver computations underlying these effects (Sec. 4.3). Sec. A.3 reports uncertainty estimates for the interface comparisons.

4.1 Escalation: Limited Rescue at High Cost

Raw escalation fails to recover most of HC's quality advantage. Raw is the default handoff interface, passing the full LC trajectory directly to HC. Across both model pairs in Panel (a) of Tab. 1, it recovers less than half of the LC–HC quality gap (QRec$=47\%$ for Claude and 36% for GPT). This limited quality recovery is common to both families; whether it also creates a severe cost premium depends on the model pair.

Raw's limited rescue comes at substantial cost. Raw escalation substantially increases cost over LC-only in both families, by approximately $4.0\times$ for Claude and $6.1\times$ for GPT. For Claude, this increase is large enough that Raw costs more than twice as much as starting with HC ($1.61 vs. $0.72). By contrast, Raw remains cheaper than HC-only for GPT ($0.36 vs. $0.47). This difference reflects the model pairs' baseline per-task economics: HC-only costs roughly $8\times$ as much as LC-only for GPT, compared with roughly $2\times$ for Claude.

For Claude, Raw's limited rescue and severe cost premium motivate a stricter question: once LC work has been incurred, is it better to continue Raw or abandon the run and restart HC? We compare Raw with two controls that charge for LC's work but discard both its conversation and working-tree edits before restarting HC from the original task state. *Abort + HC fresh* charges for LC through the switch point before running HC from scratch; *LC-full + HC-full* charges for a complete LC run

Claude (Haiku 4.5 / Opus 4.7)						**GPT-5.6** (Luna / Sol)					
Strategy	Pass	Cost	Steps	QRec	CSRet	Strategy	Pass	Cost	Steps	QRec	CSRet
(a) Escalation (LC → HC)											
LC-only	60.7	0.40	80	0	100	LC-only	58.7	0.06	14	0	100
HC-only	79.2	0.72	31	100	0	HC-only	83.7	0.47	17	100	0
Abort + HC fresh	79.2†	0.90	78	100†	−58	Abort + HC fresh	83.7†	0.51	27	100†	−11
LC-full + HC-full	79.2†	1.12	111	100†	−130	LC-full + HC-full	83.7†	0.53	31	100†	−15
Raw handoff	69.2	1.61	74	47	−285	Raw handoff	67.5	0.36	**17**	36	26
Compact$_{\text{pre}}$	71.8	**0.75**	71	60	**−11**	Compact$_{\text{pre}}$	68.8	**0.27**	18	40	**49**
Compact$_{\text{suf}}$	69.6	0.98	**67**	49	−82	Compact$_{\text{suf}}$	68.8	0.43	19	40	10
Traj-drop	**72.4**	0.81	74	**64**	−30	Traj-drop	**79.7**	0.50	25	**84**	−8
(b) Downshift (HC → LC)											
LC-only	54.6	0.41	79	0	100	LC-only	63.6	0.05	13	0	100
HC-only	75.8	0.85	35	100	0	HC-only	85.8	0.47	18	100	0
Raw handoff	65.6	**0.51**	58	50	**80**	Raw handoff	**81.0**	**0.41**	**16**	**79**	**14**
Compact$_{\text{pre}}$	**66.8**	0.52	**54**	**56**	78	Compact$_{\text{pre}}$	79.8	0.43	18	72	10
Compact$_{\text{suf}}$	63.7	0.53	65	42	73	Compact$_{\text{suf}}$	80.4	0.42	17	75	13
Traj-drop	60.9	0.59	79	28	59	Traj-drop	75.4	0.43	24	53	10

Table 1: **Aggregate coding-agent handoffs across model pairs.** Raw escalation recovers less than half of HC's quality advantage, whereas downshift offers a favorable intermediate cost–quality point. Reducing inherited LC context improves escalation, while preserving HC context improves downshift. Values are unweighted means over seven switch points on matched switched subsets; QRec and CSRet follow Sec. 3.4.

before restarting HC. Both recover HC-only quality by construction. Even after charging for the discarded LC work, both controls remain cheaper than Raw ($0.90 and $1.12, respectively, versus $1.61). Thus, for the Claude pair, Raw escalation is strictly dominated by restarting HC from scratch: restarting costs less and solves more tasks.

We next test whether reducing the inherited trajectory can improve Raw's limited rescue without discarding LC's working-tree edits.

Different trajectory reductions favor different cost–quality objectives. Both compaction interfaces yield slightly higher observed pass rates than Raw, with differences of 0.4–2.6 points (Tab. 1). Compact$_{\text{pre}}$ provides the highest cost retention among the tested escalation interfaces in both families. For Claude, it raises QRec from 47% to 60% and improves CSRet from -285% to -11%, approaching HC-only cost parity. For GPT, QRec changes only slightly, from 36% to 40%, while CSRet improves from 26% to 49%.

Taking context reduction further produces a larger quality gain. Traj-drop removes the trajectory while preserving LC's working-tree edits. With Traj-drop, QRec rises from 47% to 64% for Claude and from 36% to 84% for GPT. This quality recovery comes with negative CSRet in both families (-30% and -8%), meaning that Traj-drop remains costlier than HC-only.

Difficulty changes when Claude escalation is worthwhile. In aggregate, every Claude escalation interface costs more than HC-only, suggesting that escalation may never be worthwhile for this pair. We test whether this conclusion holds across task difficulty using the SWE-bench Verified labels. Fig. 3 reveals an exception on hard tasks.

On easy tasks, all interfaces offer a poor cost–quality trade-off, recovering only around half of the LC–HC quality gap while costing far more than HC-only (CSRet -164% to -982%; Tab. 4). In the hard-task bucket, by contrast, all three reduced-context interfaces cost less than HC-only, retaining 12–42% of LC's cost advantage while recovering 65–74% of the LC–HC quality gap. Raw does not make the same transition: it remains costlier than HC-only and recovers only 47% of the quality gap. Thus, difficulty alone does not make Raw escalation cost-effective, but it can make reduced-context escalation attractive. Because the hard subset is small ($\bar{N} \approx 24$ per cell), we treat this pattern as exploratory rather than as a cross-family finding.

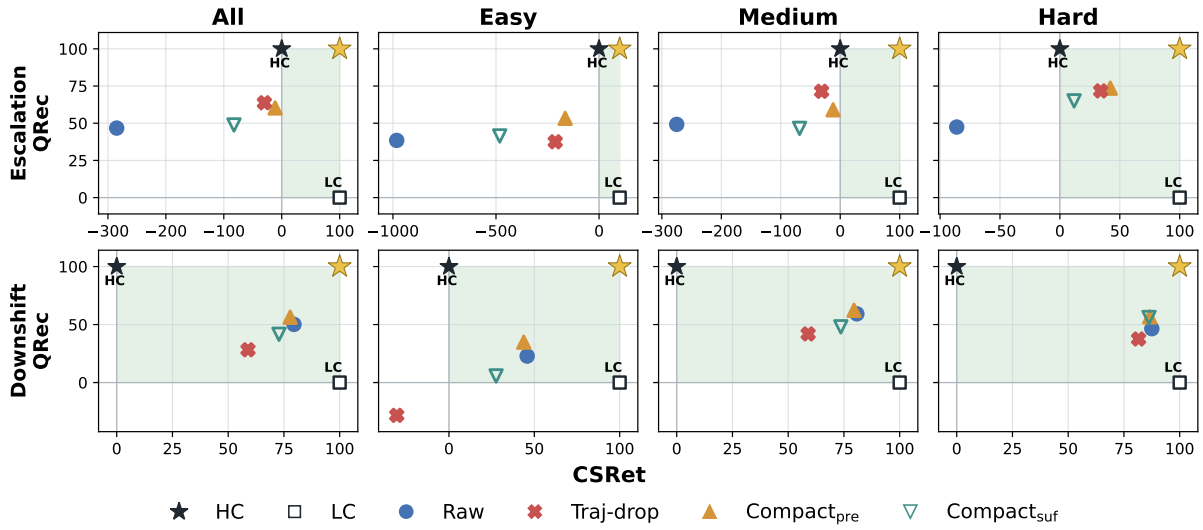

Figure 3: **Handoff effects vary with task difficulty.** For Claude, all escalation interfaces are unfavorable on easy and medium tasks, but on hard tasks reduced-context interfaces become cheaper than HC-only while recovering 65–74% of HC's quality advantage; Raw does not make this transition. In downshift, dropping the HC trajectory yields the worst cost–quality trade-off among tested interfaces in every difficulty bucket.

Takeaway. Raw escalation is a poor bargain: it recovers less than half of HC's quality advantage at several times LC's cost. For Claude, it even costs more than completing LC and then running HC from scratch. What HC inherits changes the trade-off: Compact$_{\text{pre}}$ minimizes handoff cost, while Traj-drop maximizes quality recovery.

4.2 Downshift: Context Preserves Quality

Raw downshift lets LC build on HC's work. In contrast to the limited rescue produced by Raw escalation, the same interface transfers useful HC progress when the direction reverses. Downshift tests whether LC can build on HC's partial work. For Claude, Raw HC→LC raises pass rate from 54.6% to 65.6%, while cost rises only from \$0.41 to \$0.51, retaining 80% of LC's cost advantage. GPT shows a different balance: Raw retains 79% of HC's quality advantage but only 14% of LC's cost advantage. Thus, both LC receivers convert HC's partial work into a favorable intermediate point below HC-only cost, with different balances: Claude retains most of LC's cost advantage, whereas GPT retains most of HC's quality advantage.

Removing the HC trajectory harms downshift quality. Raw downshift transfers both HC's working-tree edits and its trajectory, so its quality gain does not reveal which form of state helps LC. Traj-drop isolates the role of trajectory guidance by preserving the edits while removing the trajectory. For Claude, Traj-drop—the highest-quality escalation interface—becomes the lowest-quality and most expensive tested downshift strategy, recovering only 28% of the LC–HC quality gap, compared with 50% for Raw and 56% for Compact$_{\text{pre}}$, while retaining just 59% of LC's cost advantage. The difficulty breakdown in Fig. 3 shows that Traj-drop has the same quality-and-savings disadvantage across all difficulty levels.

The same quality reversal appears for GPT. Traj-drop recovers only 53% of the LC–HC quality gap, compared with 79% for Raw and 72–75% for the two compaction interfaces. Cost retention spans only 10–14% across the GPT interfaces, so removing the trajectory primarily harms quality rather than savings. Thus, across both model families, the HC trajectory is important for downshift quality: preserving it retains substantially more of HC's advantage than removing it entirely.

Takeaway. Downshift offers a useful operating point: LC models retain a substantial share of HC's quality advantage while remaining cheaper than HC-only. Removing the HC trajectory sharply reduces that retained quality. Together with escalation, this reveals a directional duality: dropping the sender's trajectory helps an HC receiver but harms an LC receiver.

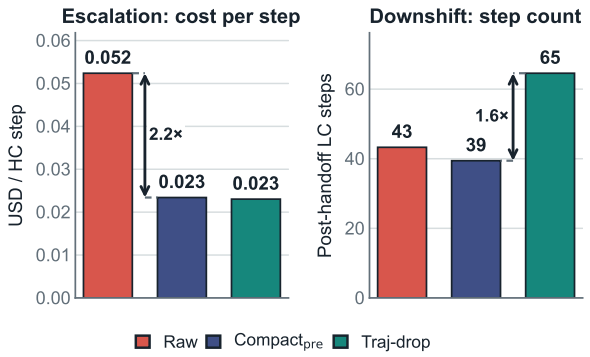

Figure 4: **The handoff tax arises through different computational mechanisms.** In Claude escalation, Raw makes each post-handoff HC step 2.2× costlier than $\text{Compact}_{\text{pre}}$; in downshift, Traj-drop makes LC take 1.6× more steps. Full LC context inflates HC calls, whereas missing HC context forces LC rework.

4.3 Cost Mechanics by Handoff Direction

Having established that trajectory removal improves escalation quality but harms downshift quality across both families, we next examine how handoff direction changes receiver-side computation and cost. We decompose post-handoff cost into receiver step count and cost per step. The Claude and GPT decompositions are shown in Fig. 4 and Appendix A.3 (Fig. 6), respectively.

Raw escalation inflates the cost of each HC step. Relative to $\text{Compact}_{\text{pre}}$, Raw raises the average post-handoff cost per HC step by 2.2× for Claude and 1.6× for GPT, while the two interfaces require a similar number of HC steps within each family. The Raw-to-$\text{Compact}_{\text{pre}}$ escalation premium therefore arises primarily from more expensive receiver calls, not from a longer HC continuation.

Traj-drop downshift takes additional LC steps. Relative to $\text{Compact}_{\text{pre}}$, Traj-drop downshift requires 1.6× the post-handoff LC steps for Claude and 2.0× for GPT, while the two interfaces have similar costs per LC step within each family. The Traj-drop-to-$\text{Compact}_{\text{pre}}$ downshift premium therefore arises primarily from additional LC work, not from more expensive calls. This is consistent with reconstructing missing HC context.

Takeaway. Raw escalation provides limited rescue, and the handoff tax depends on direction and interface. Removing inherited LC trajectory improves quality for HC receivers, whereas preserving inherited HC trajectory benefits LC receivers. The associated cost penalties manifest through different computational channels: more expensive HC steps under Raw escalation and additional LC steps under Traj-drop downshift.

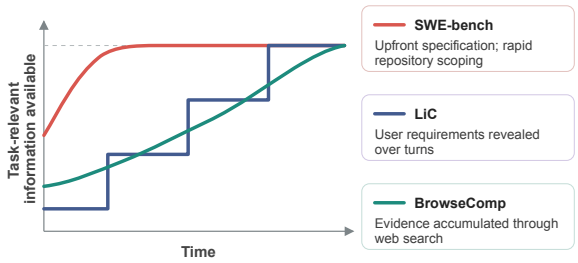

Figure 5: **Task-relevant information evolves differently across settings.** SWE-bench provides the specification upfront, followed by rapid repository scoping; LiC reveals user requirements over turns; BrowseComp accumulates evidence through web search. The schematic illustrates what information is available at different stages of execution.

Tasks	Policy	Score	Cost	QRec	CSRet
	LC-only	62.1	0.015	0	100
	HC-only	76.9	0.080	100	0
	Escalation	74.2	0.056	86	36
	Downshift	67.0	0.046	31	53

Table 2: **Late-arriving requirements favor an HC suffix.** Raw-handoff results on LiC, averaged across five task families and early, middle, and late structural switch positions for Claude.

5 Beyond the Coding-Agent Setting

Our primary study focuses on coding agents, one of the most consequential real-world settings for long-running agents, using SWE-bench, where the specification is available upfront and execution leaves persistent repository state. We next vary an orthogonal property: the task's *information dynamics*—how task-relevant information becomes available over time—which shape what the sender can accomplish before the switch and what remains for the receiver. As illustrated in Fig. 5, SWE-bench provides the specification upfront while repository information accumulates during execution; LiC (Laban et al., 2025) reveals requirements incrementally across turns; and BrowseComp (Wei et al., 2025) provides the question upfront while evidence accumulates through search. We evaluate Raw handoffs throughout to isolate this axis from interface design.

Strategy	QRec	CSRet	Steps	
			LC	HC
Full LC + full HC	100.0	−40.1	62.6	33.4
Abort + fresh HC	100.0	−8.5	23.0	33.4
Escalation	95.8	−30.0	23.0	30.4
Downshift	56.7	76.8	22.8	17.0

Table 3: **Under progressive search, escalation recovers quality but not savings.** GPT Raw-handoff results on BrowseComp, averaged across switch points.

Late-arriving requirements favor a strong suffix. We evaluate Raw handoffs on 535 LiC examples across five task families, where requirement shards progressively complete an initial high-level intent. Tab. 2 reports Claude averages across early, middle, and late handoffs; GPT and position-specific results, together with the full protocol, appear in Sec. B.

Across all five families, late-arriving requirements reverse the coding quality ordering while preserving the cost ordering: escalation recovers 86% of HC's quality advantage and retains 36% of LC's cost advantage, versus 31% and 53% for downshift. By construction, LiC remains underspecified until the final requirement shard, so the receiver is the first model able to solve the fully specified task. This favors escalation, which assigns that decisive stage to HC.

Progressive search recovers quality but not savings. We evaluate Raw handoffs on 200 BrowseComp questions filtered to require web browsing; the question is available upfront, but answer-relevant evidence accumulates through search. Tab. 3 reports GPT results. Full details in Sec. C.

For GPT, escalation nearly closes the HC quality gap (QRec $= 95.8\%$), unlike in SWE-bench, yet—as in SWE-bench—fails to produce savings: CSRet is -30.0%, worse than aborting LC and restarting HC from scratch (-8.5%). Downshift, by contrast, offers a useful middle ground, recovering 56.7% of HC's quality advantage while retaining 76.8% of LC's cost advantage. Escalation and *Abort + fresh HC* share the same 23-step LC prefix, but inheriting it shortens the HC continuation by three calls on average (30.4 vs. 33.4), suggesting that earlier search progress remains useful. Yet escalation remains substantially more expensive overall.

Takeaway. Together, these extensions show that handoff value depends on information dynamics. LiC favors an HC receiver because the task becomes solvable only after late requirements arrive; BrowseComp shows that inherited search progress can nearly recover HC quality without producing savings. Thus, whether escalation is worthwhile depends not only on model capability, but also on the task-relevant state available at handoff.

6 Discussion and Conclusion

Our results show that, under Raw handoff, escalation offers a poor cost–quality trade-off, whereas downshift yields a favorable intermediate cost–quality point. This directional asymmetry extends to interface design: reducing LC trajectory information improves escalation, whereas removing HC trajectory information harms downshift. Together, these suggest that model handoffs should be treated as a distinct inference problem rather than merely as an extension of model routing. Routing determines which model acts next, whereas the handoff interface determines what trajectory information that model inherits. In long-horizon coding agents, a handoff involves more than transferring conversational context: the receiver also inherits a repository state produced by another model. It must continue both an existing work product and a non-native trajectory, and how the trajectory is transferred alongside that persistent state can substantially affect both quality and cost. Thus, handoff design should be optimized jointly with receiver selection and switch timing.

Future directions. We use fixed switch points to isolate the consequences of a handoff from the policy that triggers it; future work can study adaptive, progress-aware policies that jointly decide when to switch and which model should continue. Similarly, we evaluate a small set of controlled interfaces to separate full trajectory transfer, compaction, and trajectory removal; future work can develop structured handoffs that selectively preserve, summarize, or discard particular trajectory components. More broadly, routing methods should treat the handoff interface as part of the switching policy rather than assume that the full trajectory simply carries forward. Evaluation should also extend across model families, capability gaps, pricing regimes, repeated rollouts, and multiple handoffs per trajectory.

Limitations

We study two model pairs (Claude Haiku 4.5 / Opus 4.7 and GPT-5.6 Luna / Sol), and our primary

coding-agent study uses one benchmark, SWE-bench Verified. Within this setting, however, we evaluate at substantial scale: 58 configurations per model family across all 500 tasks, totaling 58,000 agent runs, 2 million LLM API calls, and 36 billion processed tokens. To extend the scope of our study beyond the SWE-bench coding-agent setting, we evaluate a broader range of information dynamics but only under Raw transfer, so our interface findings are established primarily in the coding setting. Switch points are fixed in advance using model- and difficulty-calibrated percentiles of single-model trajectory length. The resulting matched switched subsets are relatively small for the hard SWE-bench stratum ($\bar{N} \approx 24$–27 per reported cell), so we treat the difficulty-conditioned findings as exploratory. In the SWE-bench study, we run a single episode for each of the 500 tasks under every configuration and therefore do not estimate variability across repeated runs of the same task under the same configuration. To preserve comparability despite this limitation, we restrict each interface comparison to the shared intersection of tasks that switched under every strategy. We also report task-clustered bootstrap confidence intervals, preserving each task's observations across switch points. Cost comparisons, difficulty analyses, and mechanism interpretations remain descriptive. Finally, dollar-cost conclusions depend on provider pricing and prompt-cache rates.

References

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. 2025. Automix: Automatically mixing language models. *Preprint*, arXiv:2310.12963.

Anthropic. 2025. Claude haiku 4.5 system card. https://www.anthropic.com/claude-haiku-4-5-system-card. Accessed: 2026-07-04.

Anthropic. 2026. Claude opus 4.7 system card. https://www.anthropic.com/claude-opus-4-7-system-card. Accessed: 2026-07-04.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *Preprint*, arXiv:2305.05176.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. *Preprint*, arXiv:2404.14618.

Raghu Vamshi Hemadri, Humaira Firdowse Mohammed, Rishabh Maheshwary, Srivatsava Daruru, Sagar Davasam, Vikas Yadav, Srinivas Sunkara, and Sai Rajeswar. 2026. R2v agent: Teaching slms when to ask for help. *Preprint*, arXiv:2605.16604.

Zoe Hitzig, Maxim Massenkoff, Eva Lyubich, Shaoyi Zhang, Ryan Heller, and Peter McCrory. 2026. Agentic coding and persistent returns to expertise.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. *Preprint*, arXiv:2403.12031.

Naman Jain, Kuan Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Dipesh KC and Anjila Budathoki. 2026. Handoff debt: The rediscovery cost when coding agents take over interrupted tasks. *Preprint*, arXiv:2606.02875.

Raad Khraishi, Iman Zafar, Katie Myles, and Greig A. Cowan. 2026. Evaluating performance drift from model switching in multi-turn llm systems. *Preprint*, arXiv:2603.03111.

Varun Kotte. 2026. Ucci: Calibrated uncertainty for cost-optimal llm cascade routing. *Preprint*, arXiv:2605.18796.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. LLMs get lost in multi-turn conversation. *Preprint*, arXiv:2505.06120.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

OpenAI. 2026. GPT-5.6: Frontier intelligence that scales with your ambition.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *Preprint*, arXiv:2305.15334.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Seongho Son, Sangwoong Yoon, Jiahua Tang, Shuhan Wang, Lorenz Wolf, and Ilija Bogunovic. 2026. Swe-router: Routing in multi-turn agentic software engineering tasks. *Preprint*, arXiv:2607.00053.

Antonios Valkanas, Soumyasundar Pal, Pavel Rumiantsev, Yingxue Zhang, and Mark Coates. 2025. C3po: Optimized large language model cascades with probabilistic cost constraints for reasoning. *Preprint*, arXiv:2511.07396.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 50208–50232. PMLR.

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, and 5 others. 2025. Openhands: An open platform for ai software developers as generalist agents. *Preprint*, arXiv:2407.16741.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *Preprint*, arXiv:2504.12516.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, and 2 others. 2025. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *Preprint*, arXiv:2412.14161.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Berkeley function calling leaderboard. `https://gorilla.cs.berkeley.edu/leaderboard.html`.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Preprint*, arXiv:2405.15793.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2024. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *Preprint*, arXiv:2310.03094.

Jiarui Zhang, Xiangyu Liu, Yong Hu, Chaoyue Niu, Hang Zeng, Shaojie Tang, Fan Wu, and Guihai Chen. 2026a. From myopic selection to long-horizon awareness: Sequential llm routing for multi-turn dialogue. *Preprint*, arXiv:2604.12385.

Yiqun Zhang, Hao Li, Zihan Wang, Shi Feng, Xiaocui Yang, Daling Wang, Bo Zhang, Lei Bai, and Shuyue

Hu. 2026b. Mtrouter: Cost-aware multi-turn llm routing with history-model joint embeddings. *Preprint*, arXiv:2604.23530.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 396–411, Online. Association for Computational Linguistics.

A SWE-bench Verified

A.1 Experimental Details

We provide the full SWE-bench Verified protocol, including the agent environment, handoff implementation and prompts, switch-point calibration, cost accounting, and software configuration.

Agent and task environment. We implement handoff by extending mini-SWE-agent with a wrapper containing exactly two models: a prefix model M_{pre} and a suffix model M_{suf}. A single agent loop and Docker environment are retained throughout the run. Consequently, the repository state, including all edits made by M_{pre}, persists after handoff; the interfaces differ only in the trajectory information supplied to M_{suf}. All models receive the same SWE-bench task prompt and the same Bash tool, and command outputs are returned to the model as tool observations. We evaluate all 500 test instances of SWE-bench Verified.

Models and handoff directions. For Claude, the low-cost (LC) and high-capability (HC) models are Claude Haiku 4.5 and Claude Opus 4.7, respectively. For GPT, they are GPT-5.6 Luna and GPT-5.6 Sol. We study both escalation, $M_{\text{LC}} \to M_{\text{HC}}$, and downshift, $M_{\text{HC}} \to M_{\text{LC}}$. Claude models were accessed through LiteLLM's chat-completions interface, whereas GPT models were accessed through Bedrock's OpenAI-compatible Responses endpoint; both used the same Bash tool, with conversation histories translated into each endpoint's native tool-call format. For sampling, we set temperature to 0 for Haiku; for GPT-5.6 and Opus 4.7, we use provider defaults, with *medium* and *high* reasoning effort, respectively.

Switch rule. Let $q = 0, 1, \ldots$ index agent model calls and let s_b be the switch point for difficulty bucket b. Calls $q < s_b$ use M_{pre}, and call $q = s_b$ is the first call to M_{suf}. Thus, s_b prefix actions and their resulting observations are complete before the suffix model is queried. Thresholds are fixed before the handoff runs at percentiles $p \in \{5, 10, 15, 25, 35, 45, 50\}$ of the prefix model's single-model termination-step distribution. They are computed separately for easy (< 15 minute), medium (15 minutes–1 hour), and hard (> 1 hour) tasks; the two longest SWE-bench difficulty categories are merged into the hard bucket. Tab. 11 report the exact thresholds.

Handoff interfaces. Let $\mathcal{T}_{1:K}$ denote the trajectory produced by the prefix model up to step K, and let $\mathcal{W}_K$ denote the persistent working tree at that point. All interfaces preserve $\mathcal{W}_K$; they differ only in the trajectory information passed to the suffix model. We compare four principal interfaces:

Raw. The suffix model receives the full trajectory $\mathcal{T}_{1:K}$ together with $\mathcal{W}_K$: the system prompt, task, prefix reasoning, tool calls, and tool observations are retained verbatim. No message announces the model change.

Compact$_{\text{pre}}$. The prefix model reads $\mathcal{T}_{1:K}$ and writes a plain-text continuation summary. The suffix model receives only this summary together with $\mathcal{W}_K$.

Compact$_{\text{suf}}$. The suffix model reads $\mathcal{T}_{1:K}$ and writes the continuation summary itself, allowing us to isolate which model should author the handoff representation. It then continues from this summary together with $\mathcal{W}_K$.

Traj-drop. No trajectory information from $\mathcal{T}_{1:K}$ is transferred, and no summarization call is made. The suffix model receives only the system prompt, original task, a static continuation message, and $\mathcal{W}_K$.

For both compact interfaces, the summarizer is given the full prefix trajectory followed by:

Compaction instruction

```
You are summarizing a coding agent trajectory
for handing off work to another coding agent.

The conversation above is a coding agent's
work-in-progress trajectory on a task.

Write a summary of what the agent did so
a different agent—with no access to this
conversation—can continue the task.
```

Let $\widehat{\mathcal{T}}_{1:K}$ denote the resulting summary. We then replace the live context with

$$[\, m_{\text{system}},\, m_{\text{task}},\, m_{\text{handoff}}(\widehat{\mathcal{T}}_{1:K}) \,],$$

where m_{handoff} uses the following message:

Compaction continuation message

```
Below is a summary of a previous agent's work
on this task. Any file changes it made are
still in the working tree.

«SUMMARY»
```

Here, «SUMMARY» is replaced by $\widehat{\mathcal{T}}_{1:K}$. The suffix model then produces its first executable action. Traj-drop uses the same three-message structure but replaces the summary with the following fixed continuation message:

Traj-drop continuation message

```
A previous agent worked on this task. Any file
changes it made are still in the working tree.

Continue solving the task.
```

The removed trajectory is written to an audit artifact but is never exposed to the suffix model. The restart controls are composed from existing matched runs rather than executed as separate trajectories. *Abort + HC fresh* combines the observed LC prefix cost and steps through K with the outcome, cost, and steps of the matched HC-only run. *LC-full + HC-full* similarly combines the cost and steps of the complete LC-only and HC-only runs. In both controls, quality is given by the matched HC-only outcome.

Accounting. Each response is tagged with its active model, zero-indexed call number, prefix/suffix phase, effective threshold, and task difficulty. The compact interfaces make one paid summarization call; its cost and usage are attributed to the handoff and folded into the first suffix step, although it is not counted as an additional environment-action step. Traj-drop and Raw introduce no additional model calls. Costs are reconstructed from provider-reported input, cache, and output-token usage using the corresponding cache-aware price cards. Output-token counts include reasoning tokens. Billable malformed responses are retained and counted. Runs use a 150-step cap and are audited so that prefix, handoff, and suffix costs reconcile exactly with total instance cost. Final patches are scored with the official SWE-bench evaluator.

Pricing and audit. Token prices follow LiteLLM v1.83.14's model-pricing registry for the configured Bedrock model identifiers, including cache pricing and cross-region premiums where applicable. The GPT-5.6 Bedrock identifiers use a pinned LiteLLM-compatible local price card distributed with our experiment configuration. The accompanying artifact records the exact invocation identifiers, price-card entries, and implementation and evaluator revisions. Before reporting results, we verify that every required task–condition run has an audited terminal trajectory and evaluator outcome. Infrastructure failures are rerun rather than scored, and no task or condition is excluded. Aggregate retry-attempt counts were not retained.

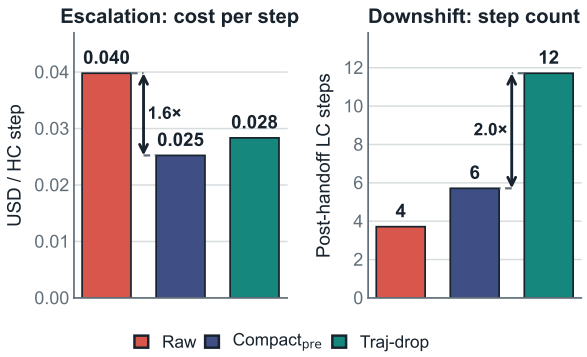

Figure 6: **GPT exhibits the same handoff-cost accounting pattern as Claude.** Relative to Compact$_{\text{pre}}$, each post-handoff HC step under Raw costs $1.6\times$ as much, while Traj-drop downshift takes $2.0\times$ as many post-handoff LC steps.

A.2 Switch-Point Calibration

Tab. 11 reports the exact difficulty-calibrated switch steps used in the SWE-bench experiments for both model families and handoff directions. Each threshold is the corresponding percentile of the prefix model's single-model termination-step distribution within a difficulty bucket. Escalation therefore uses thresholds calibrated from LC trajectories, whereas downshift uses thresholds calibrated from HC trajectories.

A.3 Additional Results and Analyses

Full results across model families. Tab. 1 reports the complete aggregate results for the Claude and GPT model pairs on matched switched subsets. Across both families, Raw escalation recovers less than half of the LC–HC quality gap, whereas downshift occupies a favorable intermediate cost–quality point. Reducing LC trajectory information improves escalation quality, while preserving HC trajectory information improves downshift quality. The analyses below disaggregate and test these aggregate patterns.

Difficulty breakdown. Tab. 4 conditions the Claude results on the SWE-bench Verified difficulty buckets, providing the numbers behind Fig. 3. Escalation economics improve with difficulty, with the structured interfaces recovering more quality at a more favorable cost. In downshift, the context-preserving interfaces consistently outperform Traj-

drop in quality recovery and cost efficiency. Results for the small hard-task subset should be interpreted directionally.

Early and late switch breakdown. Tabs. 5 and 6 split the Claude and GPT results into Early (p5–p15) and Late (p25–p50) switch windows. The central interface reversal is stable across both timing windows and model families: Traj-drop achieves the highest quality recovery in escalation and the lowest in downshift. Switch timing changes the magnitude, but not the direction, of this effect. Later escalation generally recovers less quality, whereas later downshift improves quality recovery for Raw and both compaction interfaces while retaining less savings. The finer ordering among the context-preserving interfaces varies, particularly for GPT.

Receiver-side computation. Figure 6 reports the GPT decomposition corresponding to the Claude analysis in Sec. 4.3. The same pattern holds across model families. In escalation, Raw increases the average cost of each post-handoff HC step relative to $\text{Compact}_{\text{pre}}$, while requiring a similar number of HC steps. In downshift, Traj-drop requires substantially more post-handoff LC steps than $\text{Compact}_{\text{pre}}$, while the interfaces have similar costs per LC step. Thus, the Raw escalation premium arises primarily from more expensive receiver calls, whereas the Traj-drop downshift premium arises primarily from additional LC work.

Model-change disclosure ablation. In raw handoff, a natural question is whether simply *telling* the receiver that the preceding work was produced by a different model recovers part of the escalation tax. To isolate the effect of model-change disclosure from context transformation, we compare Raw escalation with a disclosure-only variant using Claude. Both conditions preserve the full trajectory without summarization; the disclosure condition appends a single user message immediately before the suffix model's first call:

Handoff disclosure — Injected user message

```
The preceding assistant messages contain
a previous agent's work on this task. The
previous agent used a different model. Any file
changes it made are still in the working tree.
Continue solving the original task.
```

Across p5–p50, averaged uniformly over switch points on the shared fired intersection—here recomputed to include the disclosure condition, so the baselines differ slightly from Tab. 1—disclosure provides a modest quality gain at a slightly higher cost (Tab. 7). It remains below HC-only and is dominated in both quality and cost by $\text{Compact}_{\text{pre}}$ and Traj-drop. This disclosure-only prompt does not eliminate the Raw handoff tax in Claude escalation.

Strategy	Pass (%)	Cost ($)	Steps (#)	QRec (%)	CSRet (%)
LC-only	60.6	0.41	80	0	100
HC-only	78.8	0.72	31	100	0
Raw handoff	69.3	1.62	74	48	−285
Raw + disclosure	70.5	1.67	75	55	−300
$\text{Compact}_{\text{pre}}$	71.5	**0.75**	71	60	**−10**
$\text{Compact}_{\text{suf}}$	69.6	0.98	**67**	50	−83
Traj-drop	**72.1**	0.94	77	**64**	−72

Table 7: **Escalation (LC → HC), averaged over all switch points.** Intersection-only results average p5–p50 unweighted. **QRec** is quality-gap recovery and **CSRet** is cost-savings retention; both are 100% at the ideal corner. **LC**=Haiku 4.5 and **HC**=Opus 4.7. $\text{Compact}_{\text{pre}}$/$\text{Compact}_{\text{suf}}$ use the handoff-from/handoff-to model to summarize. Raw + disclosure retains the full trajectory and appends one model-change notice at the first HC call. Best Pass/QRec among handoff strategies and best Cost/Steps/CSRet among all non-baseline methods are in **bold**.

Statistical uncertainty. We quantify task-to-task variation in pairwise interface pass-rate differences. At each switch point, differences are computed on the shared subset of tasks that reached the handoff under all four interfaces. The seven switch-point differences are then averaged with equal weight, matching the aggregation in the main table.

For each model family and direction, we collect the unique task IDs appearing in at least one of the seven matched subsets. These pools contain 460 escalation and 458 downshift tasks for Claude, and 411 escalation and 426 downshift tasks for GPT. Each bootstrap replicate draws the same number of task IDs with replacement. When a task is drawn more than once, it receives the same weight under all interfaces and at every switch point where it appears. We then recompute the seven pass-rate differences and their equal-weight average. We report pointwise 95% confidence intervals from 10,000 bootstrap replicates.

Tab. 8 presents the central Traj-drop–Raw contrasts first within each direction, followed by Traj-

(a) Escalation

Strategy	Difficulty	$\bar{N}$	Pass (%)	Δ_{LC} (pp)	Δ_{HC} (pp)	QRec (%)	CSRet (%)
Raw	Easy	129	82.1	+3.9	-6.0	38	-982
	Medium	160	63.6	+10.8	-11.7	49	-275
	Hard	24	32.5	+19.2	-21.4	47	-86
$\mathrm{Compact}_{\mathrm{pre}}$	Easy	129	83.4	**+5.2**	-4.7	**53**	**-164**
	Medium	160	66.1	+13.3	-9.2	59	**-12**
	Hard	24	42.7	**+29.5**	-11.2	**74**	**42**
$\mathrm{Compact}_{\mathrm{suf}}$	Easy	129	82.2	+4.1	-5.8	41	-482
	Medium	160	63.1	+10.3	-12.2	47	-68
	Hard	24	39.6	+26.4	-14.3	65	12
Traj-drop	Easy	129	81.9	+3.7	-6.2	38	-213
	Medium	160	68.6	**+15.9**	-6.7	**71**	-31
	Hard	24	42.3	+29.0	-11.6	72	34

(b) Downshift

Strategy	Difficulty	$\bar{N}$	Pass (%)	Δ_{LC} (pp)	Δ_{HC} (pp)	QRec (%)	CSRet (%)
Raw	Easy	113	78.0	+3.0	-7.6	23	**46**
	Medium	157	61.8	+15.2	-9.8	59	**81**
	Hard	27	35.6	+20.6	-22.9	46	**88**
$\mathrm{Compact}_{\mathrm{pre}}$	Easy	113	79.0	**+4.0**	-6.6	**35**	44
	Medium	157	62.5	**+15.8**	-9.2	**62**	80
	Hard	27	40.4	**+25.4**	-18.1	**57**	87
$\mathrm{Compact}_{\mathrm{suf}}$	Easy	113	76.2	+1.2	-9.4	6	28
	Medium	157	59.1	+12.4	-12.5	48	74
	Hard	27	38.8	+23.7	-19.8	56	86
Traj-drop	Easy	113	72.8	-2.2	-12.8	-28	-31
	Medium	157	57.4	+10.7	-14.2	42	59
	Hard	27	31.3	+16.2	-27.2	38	82

Table 4: **Difficulty-conditioned handoff results.** Claude intersection-only results averaged uniformly over switch points p5–p50, split by SWE-bench Verified difficulty. Δ_{LC} is the pass-rate gain over LC-only; Δ_{HC} is the pass-rate difference from HC-only, with negative values indicating remaining gap. QRec and CSRet follow Eqs. 1–2. Bold marks the best Δ_{LC}, QRec, and CSRet within each difficulty bucket and direction.

drop–compaction contrasts and secondary comparisons among the context-preserving interfaces. The central reversal is consistent across both families: Traj-drop improves escalation pass rate and reduces downshift pass rate relative to Raw, with all four pointwise intervals excluding zero. The broader Traj-drop–compaction contrasts follow the same directional pattern, whereas comparisons among Raw and the two compaction interfaces vary across families. In particular, we do not treat summary authorship as a cross-family finding.

B Lost in Conversation

B.1 Dataset and Protocol

Corpus and task families. We build on the sharded-instruction corpora of *Lost in Conversation* (LiC) (Laban et al., 2025), which decompose each fully specified task into an ordered list of requirement *shards*, revealed one per user turn. We use five task families: **Code** (45 HumanEval (Chen et al., 2021) and 55 LiveCodeBench (Jain et al., 2024); n=100), **Database** (Spider (Yu et al., 2018); n=107), **Actions** (the parallel category of the Berkeley Function-Calling Leaderboard (Yan et al., 2024); n=105), **Math** (GSM8K (Cobbe et al., 2021); n=103), and **Data-to-text** (ToTTo (Parikh et al., 2020); n=120), for 535 tasks total. We exclude the released summarization family because its official evaluator is an external LLM judge. Tasks contain $N \in [3, 12]$ shards, with per-family medians of 4–7.

Deterministic sharded protocol. The original LiC protocol uses an LLM user simulator to paraphrase and schedule shards. We instead reveal one released shard per user turn, verbatim and in the official order, with one assistant call after every shard and no early termination. The final user turn appends a fixed family-specific instruction requesting the official answer format; only the final assistant response is evaluated. Every episode therefore contains exactly N assistant turns, and its user-turn sequence is byte-identical across models, policies, and switch points.

Models and conversations. We evaluate the same two within-family pairs as in the coding study: Claude Haiku 4.5/Opus 4.7 and GPT-5.6 Luna/Sol, with the same model identifiers and inference settings as in the coding study. LiC is chat-only and uses no tools. Assistant turns are re-fed as plain visible text without hidden reasoning blocks, making the inherited transcript model-agnostic. Each task–policy cell contains one episode; the schedule and evaluators are deterministic, while model generation is not.

Raw handoff and switch timing. At switch point K, the sender answers shards $1, \ldots, K$, and the receiver answers shards $K+1, \ldots, N$ in the same conversation. The receiver inherits the full visible history, including the sender's responses, and no message announces the model change. Because episode length is fixed, switches fire on every task. We use **Early**, $K = 1$; **Middle**, $K = \lfloor N/2 \rfloor$; and **Late**, $K = N - 1$.

Claude (Haiku 4.5 / Opus 4.7)						**GPT-5.6** (Luna / Sol)					
Strategy	Pass	Cost	Steps	QRec	CSRet	Strategy	Pass	Cost	Steps	QRec	CSRet
All											
LC-only	60.7	0.40	80	0	100	LC-only	58.7	0.06	14	0	100
HC-only	79.2	0.72	31	100	0	HC-only	83.7	0.47	17	100	0
Abort + HC fresh	79.2†	0.90	78	100†	−58	Abort + HC fresh	83.7†	0.51	27	100†	−11
LC-full + HC-full	79.2†	1.12	111	100†	−130	LC-full + HC-full	83.7†	0.53	31	100†	−14
Raw handoff	69.2	1.61	74	47	−285	Raw handoff	67.5	0.36	**17**	36	27
Compact$_{\text{pre}}$	71.8	**0.75**	71	60	**−11**	Compact$_{\text{pre}}$	68.8	**0.27**	18	40	**49**
Compact$_{\text{suf}}$	69.6	0.98	**67**	49	−82	Compact$_{\text{suf}}$	68.8	0.43	19	40	10
Traj-drop	**72.4**	0.81	74	**64**	−30	Traj-drop	**79.7**	0.50	25	**84**	−8
Early											
LC-only	63.5	0.36	73	0	100	LC-only	63.2	0.05	13	0	100
HC-only	80.4	0.64	29	100	0	HC-only	83.7	0.42	17	100	0
Abort + HC fresh	80.4†	0.78	65	100†	−49	Abort + HC fresh	83.7†	0.46	25	100†	−10
LC-full + HC-full	80.4†	1.00	102	100†	−129	LC-full + HC-full	83.7†	0.47	29	100†	−13
Raw handoff	73.1	1.38	64	57	−265	Raw handoff	72.3	0.32	**16**	44	27
Compact$_{\text{pre}}$	74.7	**0.66**	60	66	**−6**	Compact$_{\text{pre}}$	71.7	**0.24**	16	42	**49**
Compact$_{\text{suf}}$	73.6	0.82	**56**	60	−65	Compact$_{\text{suf}}$	71.4	0.37	17	40	15
Traj-drop	**75.0**	0.73	62	**69**	−32	Traj-drop	**81.2**	0.45	23	**87**	−8
Late											
LC-only	58.6	0.44	84	0	100	LC-only	55.4	0.07	15	0	100
HC-only	78.3	0.77	34	100	0	HC-only	83.8	0.50	18	100	0
Abort + HC fresh	78.3†	0.99	87	100†	−66	Abort + HC fresh	83.8†	0.55	28	100†	−12
LC-full + HC-full	78.3†	1.21	118	100†	−131	LC-full + HC-full	83.8†	0.56	32	100†	−15
Raw handoff	66.4	1.78	82	39	−299	Raw handoff	63.9	0.39	**18**	30	26
Compact$_{\text{pre}}$	69.6	**0.82**	79	56	**−16**	Compact$_{\text{pre}}$	66.6	**0.29**	18	39	**49**
Compact$_{\text{suf}}$	66.6	1.09	**74**	40	−95	Compact$_{\text{suf}}$	66.9	0.47	20	41	7
Traj-drop	**70.4**	0.87	82	**60**	−28	Traj-drop	**78.6**	0.53	27	**82**	−8

Table 5: **Coding-agent escalation by switch-timing window.** Columns compare model families. *All* averages p5–p50, *Early* averages {p5,p10,p15}, and *Late* averages {p25,p35,p45,p50}. Values are unweighted means over switch points on matched switched subsets. Pass is in percent, Cost in dollars, and Steps is the mean trajectory length. QRec is recovered LC–HC quality gap; CSRet is retained cost savings, with negative values costlier than HC-only. Gray rows are single-model baselines and restart references. Bold marks the best available handoff quality and best non-baseline cost/efficiency within each family and timing window.

Claude (Haiku 4.5 / Opus 4.7)						**GPT-5.6** (Luna / Sol)					
Strategy	Pass	Cost	Steps	QRec	CSRet	Strategy	Pass	Cost	Steps	QRec	CSRet
All											
LC-only	54.6	0.41	79	0	100	LC-only	63.6	0.05	13	0	100
HC-only	75.8	0.85	35	100	0	HC-only	85.8	0.47	18	100	0
Raw handoff	65.6	**0.51**	58	50	**80**	Raw handoff	**81.0**	**0.42**	**16**	**79**	**14**
$\text{Compact}_{\text{pre}}$	**66.8**	0.52	**54**	**56**	78	$\text{Compact}_{\text{pre}}$	79.8	0.43	18	72	10
$\text{Compact}_{\text{suf}}$	63.7	0.53	65	42	73	$\text{Compact}_{\text{suf}}$	80.4	0.42	17	75	13
Traj-drop	60.9	0.59	79	28	59	Traj-drop	75.4	0.43	24	53	10
Early											
LC-only	62.5	0.36	72	0	100	LC-only	66.5	0.05	12	0	100
HC-only	80.3	0.66	30	100	0	HC-only	86.0	0.41	17	100	0
Raw handoff	69.0	**0.41**	55	35	**83**	Raw handoff	**82.3**	0.34	**15**	**81**	20
$\text{Compact}_{\text{pre}}$	**71.0**	**0.41**	**49**	**47**	**83**	$\text{Compact}_{\text{pre}}$	79.9	0.36	17	69	16
$\text{Compact}_{\text{suf}}$	67.6	0.44	60	28	75	$\text{Compact}_{\text{suf}}$	79.9	**0.34**	16	69	**21**
Traj-drop	65.9	0.48	72	18	59	Traj-drop	77.2	0.35	22	55	18
Late											
LC-only	48.6	0.45	85	0	100	LC-only	61.4	0.06	14	0	100
HC-only	72.4	0.98	40	100	0	HC-only	85.7	0.51	19	100	0
Raw handoff	63.1	**0.58**	60	61	**77**	Raw handoff	79.9	**0.47**	**17**	77	**9**
$\text{Compact}_{\text{pre}}$	**63.6**	0.59	**58**	**63**	74	$\text{Compact}_{\text{pre}}$	79.8	0.49	19	75	5
$\text{Compact}_{\text{suf}}$	60.9	0.60	69	52	71	$\text{Compact}_{\text{suf}}$	**80.7**	0.48	18	**80**	8
Traj-drop	57.1	0.68	85	36	59	Traj-drop	74.0	0.50	26	52	4

Table 6: **Coding-agent downshift by switch-timing window.** Columns compare model families. *All* averages p5–p50, *Early* averages {p5,p10,p15}, and *Late* averages {p25,p35,p45,p50}. Values are unweighted means over switch points on matched switched subsets. Pass is in percent, Cost in dollars, and Steps is the mean trajectory length. QRec is recovered LC–HC quality gap; CSRet is retained cost savings. Gray rows are single-model baselines. Bold marks the best available handoff quality and best non-baseline cost/efficiency within each family and timing window.

Direction	Comparison		Claude		GPT	
	First	Second	ΔPass	95% CI	ΔPass	95% CI
Escalation	Traj-drop	Raw	$+3.1$	$[+1.3, +5.0]$	$+12.2$	$[+9.2, +15.4]$
	Traj-drop	$\text{Compact}_{\text{pre}}$	$+0.6$	$[-1.1, +2.3]$	$+10.9$	$[+7.5, +14.5]$
	Traj-drop	$\text{Compact}_{\text{suf}}$	$+2.8$	$[+1.0, +4.6]$	$+10.9$	$[+7.7, +14.2]$
	$\text{Compact}_{\text{pre}}$	Raw	$+2.5$	$[+0.9, +4.2]$	$+1.3$	$[-1.7, +4.3]$
	$\text{Compact}_{\text{suf}}$	Raw	$+0.4$	$[-1.2, +1.9]$	$+1.3$	$[-1.4, +4.0]$
	$\text{Compact}_{\text{pre}}$	$\text{Compact}_{\text{suf}}$	$+2.2$	$[+0.7, +3.7]$	-0.0	$[-2.4, +2.4]$
Downshift	Traj-drop	Raw	-4.7	$[-7.3, -2.1]$	-5.6	$[-8.6, -2.6]$
	Traj-drop	$\text{Compact}_{\text{pre}}$	-5.9	$[-8.5, -3.4]$	-4.4	$[-7.1, -1.8]$
	Traj-drop	$\text{Compact}_{\text{suf}}$	-2.9	$[-5.3, -0.4]$	-5.0	$[-7.8, -2.3]$
	$\text{Compact}_{\text{pre}}$	Raw	$+1.2$	$[-0.6, +3.0]$	-1.1	$[-3.2, +1.0]$
	$\text{Compact}_{\text{suf}}$	Raw	-1.9	$[-4.0, +0.3]$	-0.6	$[-2.7, +1.6]$
	$\text{Compact}_{\text{pre}}$	$\text{Compact}_{\text{suf}}$	$+3.0$	$[+1.1, +5.1]$	-0.6	$[-2.4, +1.3]$

Table 8: **Pairwise interface pass-rate differences by model family.** ΔPass is the first strategy minus the second, in percentage points. Within each direction, rows present the central Traj-drop–Raw reversal, broader Traj-drop–compaction comparisons, and secondary comparisons among context-preserving interfaces, in that order. Intervals are pointwise 95% task-clustered bootstrap confidence intervals from 10,000 resamples.

Evaluation and metrics. We use the official LiC evaluators: numeric exact match for Math, the official AST checker for Actions, execution match against the Spider test-suite databases (Zhong et al., 2020) for Database, sandboxed reference tests for Code, and multi-reference sacreBLEU (Post, 2018) for Data-to-text. "Score" is pass rate times 100 except for Data-to-text, where it is mean per-task BLEU times 100.

QRec and CSRet use matched single-model anchors under the same sharded schedule. Costs sum sender-priced prefix and receiver-priced suffix calls, including cache reads and writes. In disaggregated results, QRec is omitted whenever the absolute LC–HC anchor gap is below five points, where the normalized denominator is unstable. This rule omits Claude Math and GPT Database and Math. Aggregate QRec includes all five task families.

B.2 Results by Switch Position

Tab. 9 disaggregates the Claude results by structural switch position. Later escalation retains more savings, while quality is mostly stable. Later downshift generally improves quality while retaining less savings.

Tab. 10 reports the GPT replication. The timing–direction interaction is similar: later escalation retains more savings, whereas later downshift retains less. The main departure is Actions downshift, where the small LC–HC anchor gap makes QRec volatile and the handoff falls below LC-only at the middle and late positions.

C BrowseComp

Benchmark and task selection. We use BrowseComp (Wei et al., 2025), a benchmark of 1,266 difficult information-seeking questions with exact reference answers. To focus on tasks that require browsing, we first sample 300 questions stratified by topic. We remove 22 questions that a prior Claude Haiku–Opus screen answered successfully without browsing, then select a fixed cohort of 200 questions, again stratified by topic, from the remaining 278. Because the screening used the Claude pair, we do not claim that the resulting cohort is specifically decontaminated for the GPT-5.6 models evaluated here.

Models and browsing harness. As in the coding study, we use GPT-5.6 Luna and GPT-5.6 Sol as LC and HC, respectively. Each model operates in a single-context, ReAct-style loop with web-search and URL-retrieval tools, where one assistant inference counts as one step. Episodes allow up to 100 steps, followed when needed by a browsing-disabled finalization call. Research stops at approximately 80% context utilization, preserving capacity for pending tool output and finalization.

Raw handoff and switch timing. At switch step K, the receiver inherits the sender's complete response prefix, including reasoning items, tool calls, and tool outputs. Prefix tools are not re-executed, no message announces the change, and both models share the original step budget. Switches fire only if the sender's recorded trajectory continues beyond K. As in the SWE setup, we calibrate sender-specific switch points using percentiles. The reported p25 and p50 points are $K = 15, 31$ for escalation and $K = 10, 24$ for downshift. Each result uses the switched-only subset.

Evaluation and metrics. Answers are graded with the official BrowseComp judge prompt and parsing rule, using Claude Opus 4.7 as judge; grader costs are excluded. QRec and CSRet use matched LC-only and HC-only anchors separately at each switch point and are then averaged unweighted over p25 and p50. *Full LC + full HC* pays for both complete trajectories and uses the HC outcome. *Abort + fresh HC* pays for the LC prefix followed by a fresh complete HC trajectory.

Run accounting. Infrastructure failures are retried with backoff; exhausted episodes are rerun from scratch and never scored partially. Each task–policy cell contains one successfully audited generation, so we interpret these results descriptively. Retry attempts were not retained as a separate aggregate count.

Task	Policy	Early				Middle				Late			
		Score	Cost ($)	QRec (%)	CSRet (%)	Score	Cost ($)	QRec (%)	CSRet (%)	Score	Cost ($)	QRec (%)	CSRet (%)
Code	LC-only	65.0	0.033	0	100	65.0	0.033	0	100	65.0	0.033	0	100
	HC-only	92.0	0.163	100	0	92.0	0.163	100	0	92.0	0.163	100	0
	Escalation	88.0	0.145	85	14	89.0	0.133	89	23	80.0	0.065	56	75
	Downshift	75.0	0.047	37	89	85.0	0.082	74	62	85.0	0.156	74	5
Database	LC-only	51.4	0.011	0	100	51.4	0.011	0	100	51.4	0.011	0	100
	HC-only	72.0	0.061	100	0	72.0	0.061	100	0	72.0	0.061	100	0
	Escalation	67.3	0.068	77	−12	65.4	0.049	68	25	67.3	0.024	77	73
	Downshift	50.5	0.019	−5	85	50.5	0.032	−5	58	55.1	0.050	18	22
Actions	LC-only	57.1	0.006	0	100	57.1	0.006	0	100	57.1	0.006	0	100
	HC-only	70.5	0.035	100	0	70.5	0.035	100	0	70.5	0.035	100	0
	Escalation	73.3	0.032	121	10	69.5	0.024	93	38	70.5	0.013	100	77
	Downshift	59.0	0.010	14	85	61.0	0.018	29	58	61.0	0.029	29	22
Math	LC-only	90.3	0.008	–	100	90.3	0.008	–	100	90.3	0.008	–	100
	HC-only	94.2	0.052	–	0	94.2	0.052	–	0	94.2	0.052	–	0
	Escalation	94.2	0.049	–	6	94.2	0.040	–	27	94.2	0.018	–	77
	Downshift	89.3	0.013	–	90	91.3	0.023	–	66	94.2	0.043	–	20
Data-to-text	LC-only	46.5	0.015	0	100	46.5	0.015	0	100	46.5	0.015	0	100
	HC-only	55.6	0.089	100	0	55.6	0.089	100	0	55.6	0.089	100	0
	Escalation	55.4	0.081	97	10	52.9	0.061	70	37	51.9	0.041	59	65
	Downshift	47.4	0.028	9	83	49.9	0.061	37	38	51.6	0.076	56	16

Table 9: **Raw handoff by switch position in sharded, underspecified conversations, Claude pair.** Early, middle, and late are structural switch positions. Later escalation retains more savings, while quality is mostly stable unless prior turns accumulate deliverable state. Later downshift generally improves quality while retaining less savings. Score is pass rate except for Data-to-text, which reports BLEU$\times$100.

Task	Policy	Early				Middle				Late			
		Score	Cost ($)	QRec (%)	CSRet (%)	Score	Cost ($)	QRec (%)	CSRet (%)	Score	Cost ($)	QRec (%)	CSRet (%)
Code	LC-only	85.0	0.017	0	100	85.0	0.017	0	100	85.0	0.017	0	100
	HC-only	95.0	0.057	100	0	95.0	0.057	100	0	95.0	0.057	100	0
	Escalation	96.0	0.052	110	13	94.0	0.037	90	49	94.0	0.024	90	82
	Downshift	90.0	0.024	50	82	94.0	0.037	90	50	96.0	0.052	110	13
Database	LC-only	65.4	0.008	–	100	65.4	0.008	–	100	65.4	0.008	–	100
	HC-only	61.7	0.027	–	0	61.7	0.027	–	0	61.7	0.027	–	0
	Escalation	64.5	0.025	–	14	65.4	0.019	–	44	66.4	0.014	–	70
	Downshift	60.7	0.012	–	80	59.8	0.017	–	52	64.5	0.024	–	18
Actions	LC-only	73.3	0.005	0	100	73.3	0.005	0	100	73.3	0.005	0	100
	HC-only	79.0	0.028	100	0	79.0	0.028	100	0	79.0	0.028	100	0
	Escalation	82.9	0.023	167	23	79.0	0.017	100	50	82.9	0.009	167	83
	Downshift	73.3	0.011	0	76	68.6	0.017	−83	49	70.5	0.023	−50	23
Math	LC-only	92.2	0.005	–	100	92.2	0.005	–	100	92.2	0.005	–	100
	HC-only	94.2	0.024	–	0	94.2	0.024	–	0	94.2	0.024	–	0
	Escalation	94.2	0.022	–	10	92.2	0.018	–	32	96.1	0.010	–	76
	Downshift	89.3	0.007	–	91	91.3	0.011	–	68	91.3	0.019	–	24
Data-to-text	LC-only	33.4	0.008	0	100	33.4	0.008	0	100	33.4	0.008	0	100
	HC-only	49.9	0.036	100	0	49.9	0.036	100	0	49.9	0.036	100	0
	Escalation	49.3	0.029	96	24	49.5	0.024	98	40	45.6	0.018	74	63
	Downshift	32.9	0.009	−3	95	32.7	0.023	−4	45	37.0	0.030	22	22

Table 10: **Raw handoff by switch position in sharded, underspecified conversations, GPT pair.** Early, middle, and late are structural switch positions. Later escalation retains more savings, while quality is mostly stable unless prior turns accumulate deliverable state. Later downshift generally improves quality while retaining less savings. Score is pass rate except for Data-to-text, which reports BLEU×100.

Prefix model	Difficulty	N	Switch step by prefix-run percentile						
			p5	p10	p15	p25	p35	p45	p50
Haiku 4.5	Easy	194	25	30	34	37	42	46	50
	Medium	261	35	41	45	51	58	63	66
	Hard	45	48	54	58	63	77	86	94
Opus 4.7	Easy	194	5	7	8	10	12	14	14
	Medium	261	8	12	13	16	18	21	23
	Hard	45	18	19	20	24	30	33	37
GPT-5.6 Luna	Easy	194	7	8	8	9	9	10	10
	Medium	261	8	8	9	10	10	11	11
	Hard	45	9	10	11	11	11	12	12
GPT-5.6 Sol	Easy	194	9	10	10	11	12	13	13
	Medium	261	10	11	11	13	13	15	15
	Hard	45	12	13	14	15	16	18	19

Table 11: **Difficulty-calibrated switch steps by prefix model.** Steps are agent turns, corresponding to model API calls. Each threshold is the specified percentile of the prefix model's single-model termination-step distribution within the given difficulty bucket.