WEIGHTS OR WORDS? WHERE TO SPEND AN AGENT'S ADAPTATION BUDGET

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent can be improved by training its weights, or by rewriting the text wrapped around it: the system prompt, the tool descriptions, and the accumulated notes that tell it how to behave, which we call its harness. The two literatures disagree about which of these is the better use of a fixed budget. Reflective prompt rewriting has been reported to beat GRPO using an order of magnitude fewer episodes; on longer-horizon agent tasks, however, the same family of methods has been reported to lose to a single hand-written prompt. Yet neither comparison holds compute fixed, and only under a fixed budget does the comparison yield a fair and usable conclusion. We therefore hold it fixed, comparing training the weights alone, rewriting the harness alone, and doing both, under identical FLOPs on a multi-turn retrieval agent. Our systematic experiments reveal that rewriting the harness does not turn budget into accuracy: a tenfold budget increase changes success by at most 0.017, and every interval contains zero. The same increase spent on the weights, by contrast, yields $+0.110$. This asymmetry is not specific to one setting: the null for rewriting reappears in a second environment and at a second model scale, whereas the advantage of doing both, the strongest configuration on retrieval, does not survive the change of environment. Nor do the two sides combine, since transplanting a rewritten harness onto separately trained weights adds nothing; they are not interchangeable stores of the same knowledge. The null itself resists the obvious explanations, as a rewriter nine times larger writes no better prompts, while a third of the proposals are accepted and read like sound advice. Taken together, these results suggest that what limits such an agent is not the quality of the harness it is given but its capacity to exploit it, which rewriting cannot supply.

1 INTRODUCTION

Language-model agents can be improved by training their weights: reinforcement learning from human feedback adapts instruction-following behaviour (Ouyang et al., 2022), while self-training can improve reasoning (Zelikman et al., 2022) and tool use (Schick et al., 2023). Parameter-efficient methods such as LoRA reduce the number of trainable parameters (Hu et al., 2021). A different route changes the text around a frozen model. ReAct illustrates how prompting can interleave reasoning and actions (Yao et al., 2022); Reflexion stores verbal feedback for subsequent agent trials (Shinn et al., 2023), while Self-Refine iteratively revises outputs using model-generated feedback (Madaan et al., 2023).

Automatic prompt optimization makes this textual route systematic. ProTeGi uses textual critiques, OPRO proposes instructions from optimization histories, and EvoPrompt applies evolutionary search (Pryzant et al., 2023; Yang et al., 2024; Guo et al., 2024). DSPy and MIPRO extend optimization to multi-stage language-model programs (Khattab et al., 2024; Opsahl-Ong et al., 2024). Here we call the system prompt, tool descriptions, and accumulated task notes the agent's *harness*. The proposal is to leave the weights alone and rewrite this harness using feedback. This has been reported to beat GRPO while consuming an order of magnitude fewer episodes (GEPA; Agrawal et al., 2026). On longer-horizon agent tasks, however, the same family of methods produces text that loses to a single hand-written prompt (Ray & Goyal, 2026). What the two reports leave open is the question that has to be answered before any compute is spent: for a fixed budget, does spending it on the weights or on the harness buy more improvement?

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.

Neither of the two reports settles it, because neither was measured on the axis the decision turns on. GEPA counts episodes, and the model doing its rewriting is a frontier model whose compute falls outside the budget being compared; the side called cheap is cheap in part because some of its cost is never priced. The negative result never places automatic rewriting on the same scale as weight training at all. The two are not even in direct conflict, since one studies prompt optimization across language-model pipelines and the other a multi-turn agent. Compute-optimal pretraining (Hoffmann et al., 2022) and test-time scaling (Snell et al., 2025) show why allocating a fixed budget is itself an empirical question. Our question is how much accuracy weight training and harness rewriting each return per unit of compute for the same multi-turn agent.

This paper measures it. We take a multi-turn retrieval agent on HotpotQA (Yang et al., 2018) and run three configurations under identical FLOPs: train the weights and leave the harness alone, rewrite the harness and leave the weights alone, or alternate between the two. We then repeat the entire comparison on a second task family and at a second model scale. Below a certain budget none of this matters: at the lower of our two budgets the three configurations fall within 0.024 of each other, and none of them beats leaving the agent alone.

Above that budget, the two sides come apart. Ten times more compute spent on the weights buys $+0.110$ in success where material is plentiful. The same increase spent on the harness buys at most $+0.017$, and every confidence interval contains zero.

This is not because the rewritten harness is poor, and not because the agent ignores it. A third of the rewrite proposals are accepted, and they contain advice a person would endorse: decompose a multi-hop question into sequential searches, issue another query when a passage names several candidates, return the exact span rather than a paraphrase. The agent then does what the harness says, issuing 1.79 searches per question where the unmodified agent issues 1.46. Its accuracy does not move. Nor is the writer too weak, since a rewriter nine times larger produces nothing measurably better.

The null is not an artefact of one task. Ten times more compute on the harness costs 0.053 on a second task family and 0.025 at a second model scale. What does not carry over is the benefit of alternating between the two sides: it is the strongest configuration on retrieval and the only one that helps when material is scarce, yet on the second task family it converts no budget at all. We report it as a property of retrieval rather than a rule.

If the harness side returns nothing on its own, its output might still be worth something in combination, on the view that prompts and weights hold different kinds of knowledge (E-SPL; Zhang et al., 2026b). They do not combine. Transplanting a rewritten harness onto separately trained weights adds nothing, and the two gains do not add. Whatever rewriting produces, it is not a portable copy of what training produces.

Contributions. This paper gives a systematic answer to whether a fixed budget buys more improvement in an agent's weights or in its harness. We measure what each side returns per unit of compute under a common budget, verifying adherence run by run, across two task families and two model scales. We show that the harness side returns nothing while the weight side returns a measurable gain, and that neither the rewriter's capability nor a failure of the rewriting loop accounts for the difference. Finally we separate what generalises from what does not: the harness side's null travels, while the advantage of alternating between the two sides is specific to retrieval.

2 Related Work

Changing the harness. GEPA (Agrawal et al., 2026) introduced automatic prompt rewriting with Pareto-front parent selection. A line of follow-ups extends it to full scaffolds and skill libraries (Wei et al., 2026; Luo et al., 2026; Wang et al., 2026a; Kulkarni et al., 2026; Bai et al., 2026; Song et al., 2026). One Recipe, Many Harnesses (Yang et al., 2026) fixes the rewriting recipe and analyses what the rewritten text encodes across languages and base models, without comparing against training the weights. On the skeptical side, Simple Baselines (Gideoni et al., 2026) finds that simple baselines match or beat elaborate rewriting in three domains including agent scaffold design. We do not propose a new rewriting method. We take the standard recipe and place it on the same budget axis as weight training.

Changing the weights. Multi-turn agentic RL is active, with much recent work on credit assignment (TRACE and related work; Tao et al., 2026; Zhang et al., 2026a; Wang et al., 2026c; Wu et al., 2026; Wang et al., 2026b). We fix credit assignment to uniform GRPO in all three configurations. Improving it is orthogonal to our question and varying it would confound the comparison.

Changing both. E-SPL is discussed in Section 1. It does not report what changing both costs relative to what it returns, which is what Section 4.4 measures.

3 METHODOLOGY

3.1 WHAT COUNTS AS THE HARNESS, AND WHAT IS HELD FIXED

A harness is a tuple (`system_prompt`, `skills`, `evidence_k`, `max_turns`). It is serializable, has a content fingerprint, and can be mutated. Rewriting may change `system_prompt` and `skills` only. The fields `evidence_k` and `max_turns` are set by the experimental condition and are overwritten in every offspring. Without this constraint the harness side could lower task difficulty by requesting more material per search, and the comparison would no longer hold difficulty fixed.

3.2 MATCHED-COMPUTE ACCOUNTING

The primary axis is FLOPs. Writing N for the parameter count of the model being improved and T_{roll}, T_{upd}, T_{rw} for the tokens consumed by rollouts, by weight updates and by rewriting, a run spends

$$F \;=\; \underbrace{2NT_{\mathrm{roll}}}_{\text{rollouts}} \;+\; \underbrace{4NT_{\mathrm{upd}}}_{\text{updates}} \;+\; \underbrace{2NT_{\mathrm{rw}}}_{\text{rewriting}}. \tag{1}$$

The factor 4 on the update term, rather than the 6 of full fine-tuning, reflects LoRA: the backward pass still traverses the frozen layers but no gradient is accumulated for them. The rewriting term is charged at inference cost and enters the same axis only when the rewriter and the model being improved are the same network, which is the condition under which Equation 1 is comparable across configurations. We also record tokens, traces, model calls, wall-clock time, and dollar cost for every run. Budget checks are *lookahead*. Let F_i be the spend after i steps, B the budget, and $\hat{c}_i = \max_{j \le i}(F_j - F_{j-1})$ the largest single-step cost observed so far. A run halts before step $i+1$ whenever

$$F_i \;\ge\; B \qquad \text{or} \qquad F_i + \hat{c}_i \;>\; B. \tag{2}$$

The second clause is the lookahead: taking $\hat{c}_i$ as a running maximum rather than the cost of the most recent step makes the test conservative, so a run under-spends rather than over-spends. Whatever it does spend enters the horizontal axis of every figure as measured, not as allotted. Observed spending is 0.52 to 1.22 times the budget given, with a median of 0.98. The deviation is not symmetric across configurations: rewriting the harness never underspends, while alternating between the two sides never overspends (Appendix B).

Runs in which the rewriting model differs from the model being improved are excluded from the FLOPs axis and compared on the token, wall-clock, and dollar axes only, with the exclusion stated wherever those numbers appear.

3.3 SEPARATING THE REWRITING LOOP FROM THE REWRITER'S CAPABILITY

In the standard configuration the rewriter is the same 4B model whose harness is being rewritten. This is the fair comparison against weight training, since both sides then have access only to the model's own capability. We add a variant in which the rewriter is a 35B model from the same family. The difference isolates *the rewriter's capability* from *the rewriting loop*.

So that a reader can judge whether rewriting is actually happening, we log and report two quantities: the acceptance rate of proposals, and the number of proposals that fail to parse. In the reported runs, 247 generations produced 84 accepted proposals, an acceptance rate of 34 percent, with 6 parse failures. A run whose first generation cannot be parsed is treated as invalid.

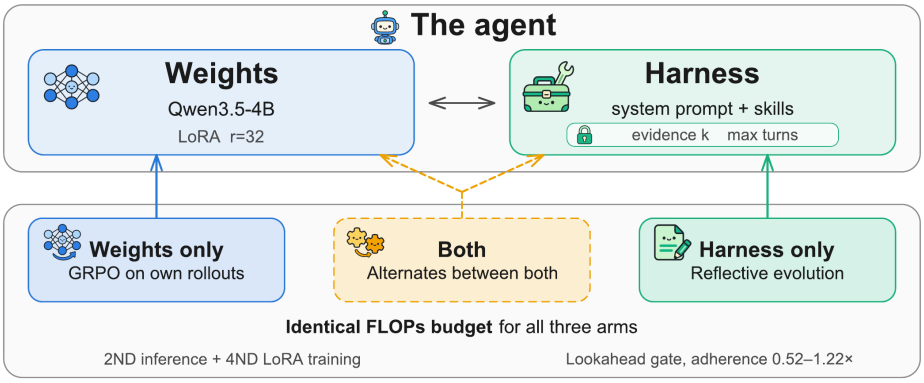


Figure 1: Experimental design. We treat the agent as two separately adaptable parts, its weights and its harness, and vary only which of the two a configuration is permitted to change. Within the harness, evidence_k and max_turns are fixed by the experimental condition and overwritten in every rewrite, so that no configuration can lower task difficulty in order to raise its own score, and all three draw on the same FLOPs budget.

3.4 What we compare

Figure 1 shows the design.

Weights only: LoRA of rank 32 trained with GRPO, harness frozen at its initial configuration. **Harness only**: automatic rewriting with Pareto-front parent selection, weights frozen. **Both**: every K GRPO steps, one generation of harness rewriting runs, with candidates evaluated under the current weights.

Controls. The *unchanged* agent uses the initial harness and base weights. The *hand-written harness* was produced by one author in about two hours after reading twenty failure traces. The *retrieval-forbidden probe* answers from parametric memory alone, which bounds how much of any score can come from memorisation rather than from search.

3.5 The exchange experiment

If prompts hold declarative knowledge and weights hold procedural knowledge, as E-SPL proposes, then the harness produced by the harness-only configuration should transfer onto weights trained by the weights-only configuration and the two gains should add. We test this with a four-cell design at fixed budget: base weights with the initial harness, base weights with the rewritten harness, trained weights with the initial harness, and trained weights with the rewritten harness. All four cells are pure inference. Write $s(W, H)$ for the success of weights W under harness H, with W_0 the base weights, W_A the trained weights, H_0 the initial harness and H_B the rewritten harness. The transfer effect and the additivity residual are

$$\tau = s(W_A, H_B) - s(W_A, H_0), \tag{3}$$

$$\rho = \big[s(W_A, H_B) - s(W_A, H_0)\big] - \big[s(W_0, H_B) - s(W_0, H_0)\big]. \tag{4}$$

A positive τ says the rewritten harness helps weights it never co-evolved with. A residual ρ near zero says the two gains simply add, which is what the division-of-labour hypothesis predicts; a negative ρ says the weights have already absorbed whatever the harness was supplying. As a precondition we also report the weight gain $s(W_A, H_0) - s(W_0, H_0)$, which must reproduce the corresponding cell of the main matrix for the four cells to be comparable at all. We report only matrices that meet this precondition, and state the check in Section 4.7.

4 Experiments

4.1 Setup

Task. Multi-hop question answering over HotpotQA distractor passages with a local BM25 retriever. Tool calls return in microseconds, so wall-clock numbers reflect model computation rather

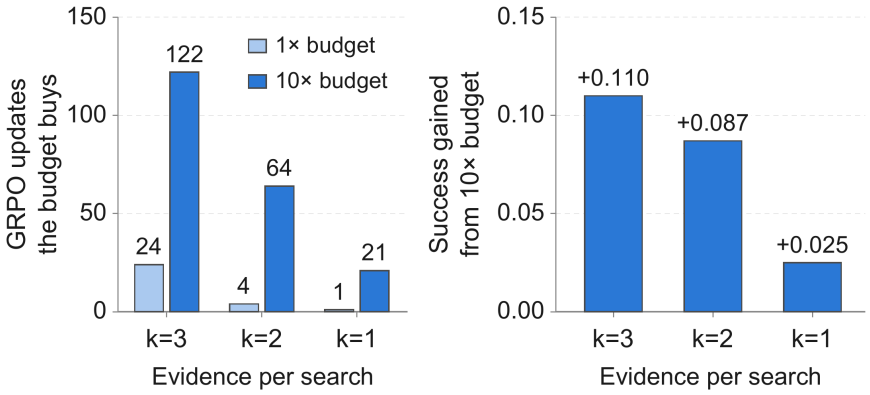


Figure 2: Left: GRPO updates each tier buys under each retrieval condition. Right: success gained from the tenfold increase. Scarcer material makes each update more expensive.

than environment latency. The agent searches, reads the returned passages, and answers within `max_turns`. Success uses a strict span-level scorer that rejects multi-candidate answers, and the same scorer provides the training reward, so reward quality is held fixed and high.

Retrieval condition. The knob `evidence_k` sets how many passages a search returns. We preregistered it as a task-length knob, expecting agents to compensate for less material with more search turns. The manipulation check refutes that reading: realised turns stay flat as k drops (2.11, 2.15, and 2.32 mean turns at $k = 3, 2, 1$). Agents answer from less material instead of searching longer. We therefore interpret k as how much material a search returns and report it as such. The unchanged baseline is not monotone in k (0.535, 0.539, 0.359), which is consistent with fewer distractor passages partly offsetting scarcer material under a strict scorer.

Budget tiers. The low tier is 4.4×10^{16} FLOPs, set as 1.15 times the measured cost of one GRPO step under the most expensive condition, so that all three configurations can complete at least one update. The high tier is ten times that. The tiers buy very different numbers of updates depending on the retrieval condition: 24, 4, and 1 updates at the low tier, and 122, 64, and 21 at the high tier (Figure 2).

Matrix. Three configurations by two tiers by three retrieval conditions by three seeds, giving 54 runs, plus 21 control probes, 6 rewriter-capability runs, and 12 cross-scale runs. The model being improved is Qwen3.5-4B; the cross-scale replication uses Qwen3.5-9B; the large rewriter is Qwen3.5-35B-A3B.

Statistics. Paired bootstrap over seeds with 10,000 resamples for all contrasts, reporting the effect and its 95 percent interval. We call a contrast significant only when the interval excludes zero.

Preregistration. Predictions, falsification conditions, and the analysis plan were frozen before the main data. Two deviations are marked where they occur: the reinterpretation of `evidence_k`, and the recalibration of the budget base against the measured cost of a single step.

4.2 At the low budget nothing beats leaving the agent alone

Figure 3 plots accuracy against compute for the three configurations. Table 1 gives the full matrix.

The three configurations fall within 0.024 of each other at the low tier, and none exceeds the unchanged baseline under any retrieval condition. The largest low-tier value, the weight side at $k = 3$, equals the unchanged score to three decimals. The headline form of the claim that cheap rewriting wins, namely that rewriting the prompt at a fraction of the RL budget already improves the agent,

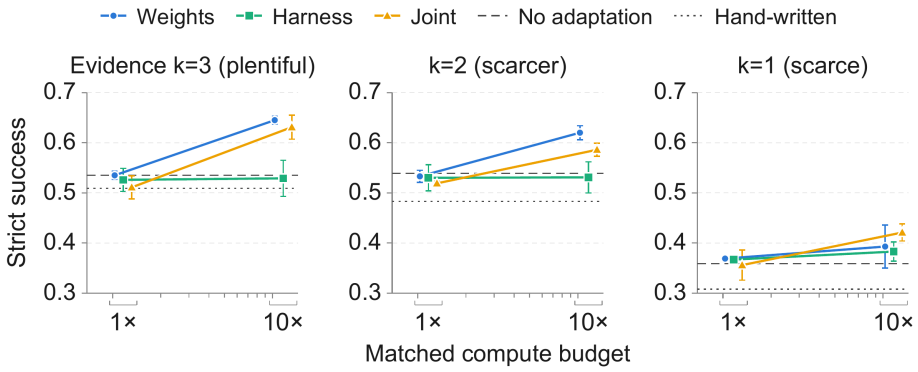


Figure 3: Accuracy against matched compute. Panels split by retrieval condition, matched budget on a log axis, three configurations with standard-deviation bands, and two reference lines. The weight side and alternating both rise with budget; the harness side is flat under every condition.

Table 1: Success under each configuration, budget and retrieval condition, averaged over three seeds and reported with the standard deviation across them. Bold marks the highest value in each column at the high budget; at the low budget no configuration leads. For reference, the table omits the controls: the unchanged agent scores 0.535, 0.539 and 0.359 as k falls from three to one, a hand-written harness scores 0.509, 0.483 and 0.308, and an agent forbidden from retrieving scores 0.053.

What is adapted	Budget	$k = 3$	$k = 2$	$k = 1$
Weights	1x	0.535 ± 0.008	0.533 ± 0.012	0.369 ± 0.005
Harness	1x	0.526 ± 0.023	0.530 ± 0.026	0.367 ± 0.008
Both	1x	0.511 ± 0.023	0.519 ± 0.004	0.356 ± 0.030
Weights	10x	$\mathbf{0.645 \pm 0.008}$	$\mathbf{0.620 \pm 0.014}$	0.393 ± 0.043
Harness	10x	0.529 ± 0.036	0.531 ± 0.031	0.383 ± 0.019
Both	10x	0.631 ± 0.024	0.586 ± 0.013	$\mathbf{0.421 \pm 0.017}$

Table 2: Each row contrasts two settings by resampling the three seeds in pairs 10,000 times and reporting the mean difference with its 95 percent interval. An asterisk marks an interval that excludes zero. The first three rows ask whether a tenfold budget increase helps a given configuration; the last two compare configurations with each other at the high budget.

Contrast	$k = 3$	$k = 2$	$k = 1$
Weights, 10x − 1x	$+0.110\,[+0.100, +0.120]^*$	$+0.087\,[+0.070, +0.105]^*$	$+0.025\,[-0.019, +0.064]$
Harness, 10x − 1x	$+0.003\,[-0.038, +0.041]$	$+0.001\,[-0.037, +0.038]$	$+0.017\,[-0.001, +0.037]$
Both, 10x − 1x	$+0.121\,[+0.091, +0.150]^*$	$+0.067\,[+0.055, +0.081]^*$	$+0.065\,[+0.031, +0.095]^*$
10x, Weights − Harness	$+0.115\,[+0.088, +0.156]^*$	$+0.089\,[+0.056, +0.119]^*$	$+0.010\,[-0.035, +0.051]$
10x, Both − Weights	$-0.013\,[-0.040, +0.006]$	$-0.034\,[-0.052, -0.016]^*$	$+0.028\,[-0.013, +0.071]$

does not appear in our data. What the low tier buys is visible in Figure 2: at $k = 3$ it affords 24 GRPO updates, at $k = 1$ only one.

4.3 Rewriting the harness does not convert budget into accuracy

The harness side is flat. A tenfold budget increase moves it by $+0.003$, $+0.001$, and $+0.017$, and all three intervals cover zero. At the high tier the weight side exceeds it by 0.115 and 0.089 under the two conditions with more material, both significant (Table 2).

The rewriting does happen. The acceptance rate is 34 percent (247 generations, 84 accepted, 6 parse failures), and the accepted proposals are task-specific strategy text. A representative accepted

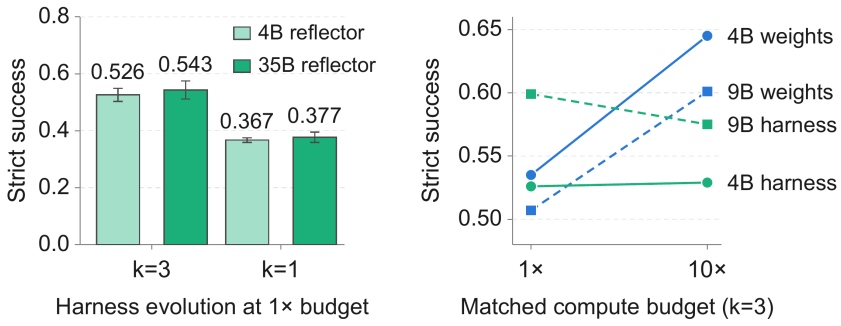


Figure 4: Left: 4B against 35B rewriter at the low tier; neither difference is significant. Right: two model scales by two configurations. The weight side rises at both scales and the harness side does not, so scale changes the intercept rather than the slope.

harness instructs the agent to break multi-hop questions into sequential steps, to disambiguate with an additional search when a passage contains several candidates, and to return the exact span rather than a paraphrase. The text is reasonable. It does not raise the score.

The harness side is also the highest-variance configuration. Its seed-level standard deviation reaches 0.036 against 0.008 for the weight side at the same cell. Across all cells, roughly half of its runs end with the initial harness as the best member of the Pareto pool, meaning rewriting produced nothing that beat its starting point. A practitioner choosing the harness side is buying a distribution with a heavy upper tail and a mode at zero gain.

4.4 Alternating between the two sides pays off on retrieval

Changing both converts budget into success under all three retrieval conditions: $+0.121$, $+0.067$, and $+0.065$, all significant. It is the worst configuration at the low tier; at the high tier it beats the weight side only where material is scarce, matches it where material is plentiful, and falls significantly behind it at the intermediate condition. Under the hardest condition, where training the weights alone does not reach significance ($+0.025$, interval covering zero), changing both does ($+0.065$). The claim of E-SPL that combining both sides is superior holds in our data at the high tier and fails at the low tier. Section 4.8 shows that it also fails to replicate in a second environment, so we state it as a property of this task rather than a general one.

The cost of each swap is measurable, and it explains the low-budget deficit. We evaluate the same weights on a fixed held-out set immediately before and after every swap of the harness. Across 20 swaps, accuracy falls by 0.031 on average (95 percent interval $[+0.003, +0.061]$), and the drop is larger at the low budget (0.063 over five swaps) than at the high budget (0.021 over fifteen). Each swap discards part of what the current weights had adapted to. At the high budget there are enough later updates to recover the loss; at the low budget there are not, which is why alternating is the weakest of the three there.

4.5 A larger rewriter writes no better prompts

Replacing the 4B rewriter with a 35B model from the same family changes success by $+0.017\,[-0.020, +0.051]$ at $k = 3$ and $+0.010\,[-0.007, +0.029]$ at $k = 1$. Neither is significant (Figure 4, left). Note the accounting: the large rewriter's own compute is not charged to the budget, so if it were, these runs would not be at the low tier at all.

This also explains how the two opposed results in the literature can both hold. When the rewriter is a frontier model and only the model being improved is charged to the budget, automatic rewriting looks efficient. When the rewriter is held at the same capability as the model being improved, it produces nothing. What remains is that the rewriting loop itself does not convert budget into success.

4.6 Replication at a second model scale

All six cells are complete and the main conclusion replicates at 9B. Spending more on the weights works: 0.507 at the low tier and 0.601 at the high tier, a gain of $+0.093\,[+0.079, +0.109]$, against $+0.110$ at 4B. Spending more on the harness does not: 0.599 and 0.575, a change of $-0.025\,[-0.061, +0.012]$ whose interval covers zero, matching the $+0.003$ we measure at 4B.

Scale changes the intercept, not the slope. At the low tier the 9B harness side is significantly above the 9B weight side ($+0.092\,[+0.065, +0.114]$), where at 4B the two differ by 0.009 and the interval covers zero. That advantage disappears at the high tier (-0.026, interval covering zero) because the weight configuration gains 0.093 while the harness side loses ground. The correct statement is that a larger model raises the standing value of a harness when nothing is trained, and that the harness side still cannot absorb budget. This makes the Section 4.3 result hold across both scales rather than restricting it to the smaller model.

4.7 The exchange experiment

Under the condition with the most material, we first check the precondition of Section 3.5: the reloaded weights gain $+0.108$ and $+0.082$ across two seeds, against $+0.110$ for the corresponding cell of Table 1. The precondition is met.

Transplanting the harness produced by the harness-only configuration onto base weights changes success by -0.002 and $+0.020$; transplanting it onto the trained weights changes success by -0.006 and $+0.024$. Additivity residuals are -0.004 and $+0.004$. Under the division-of-labour hypothesis the transfer terms should be positive and substantial. What we find is that the rewritten harness carries almost nothing onto either substrate, while the same weights alone carry $+0.108$ and $+0.082$. The division of labour proposed by E-SPL does not hold in this setting.

4.8 A second environment

We replicate the main comparison on TextWorldExpress, the venue of From Agent Failures to Text Policies (Ray & Goyal, 2026), where success is decided by the game rather than by a scorer we wrote. The three configurations run under matched FLOPs at the same two tiers, three seeds each (Figure 5, Table 3).

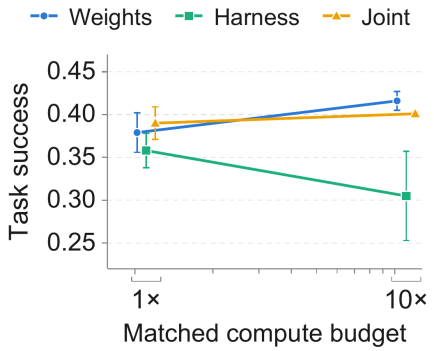


Figure 5: The second environment, plotted as in Figure 3. The harness side falls with budget, the weight side rises, and alternating both is flat.

The harness side's null reproduces and is if anything stronger: a tenfold budget increase moves it by -0.053, and at the high tier the weight side exceeds it by $+0.111\,[+0.028, +0.172]$ and alternating both by $+0.096\,[+0.020, +0.144]$, both significant. Its seed-level standard deviation at the high tier is 0.052, the largest of any cell in either environment, with two of three seeds falling well below their low-tier counterparts. Whatever the rewriting is doing here, it is not buying success, and spending more on it is if anything harmful.

Two results do not carry over. The weight side still gains but no longer significantly ($+0.037$, interval covering zero), which is consistent with our expectation that a longer episode makes each update more expensive; with three seeds and per-seed differences of -0.006, $+0.072$, and $+0.044$ the design lacks the power to resolve it. More importantly, alternating both is flat ($+0.011$, interval covering zero) and at the high tier sits 0.015 below the weight side, a small but significant deficit. On retrieval alternating both was the only one significant under all three conditions; here it converts no budget at all. We therefore report the advantage of changing both sides as a property of the retrieval task, not a general one.

What survives both the change of environment and the change of model scale is the asymmetry itself: budget spent on the weights can buy success and budget spent on the harness cannot.

Table 3: The same comparison on TextWorldExpress, averaged over three seeds and reported with the standard deviation across them, with the paired bootstrap on the tenfold budget increase in the final column. Success here is decided by the game rather than by a scorer we wrote, so absolute values are not comparable with Table 1; only the direction and size of each budget effect are.

Configuration	1x	10x	10x − 1x
Weights	0.379 ± 0.023	0.416 ± 0.011	$+0.037\,[-0.006, +0.072]$
Harness	0.358 ± 0.020	0.305 ± 0.052	$-0.053\,[-0.102, +0.012]$
Both	0.390 ± 0.019	0.401 ± 0.002	$+0.011\,[-0.002, +0.036]$

5 Discussion

A selection rule. The results compress into three lines. If the available budget is below the cost of convergence, none of the three configurations is worth funding: they fall within 0.024 of each other, none beats leaving the agent alone, and a hand-written harness is no better than leaving it alone either. If the budget is large and a search returns plenty of material, train the weights. If the budget is large but material is scarce, the only configuration that pays off in our retrieval data is training the weights and rewriting the harness together, but that advantage does not survive the move to a second environment, so we would not rely on it outside a task of the same shape.

Why the field disagreed. The two results we set out to reconcile were both correct about their own setup. Automatic rewriting looks efficient when its rewriter is a frontier model and only the model being improved is charged to the budget. A rewriter held at matched capability produces nothing, which is what Ray & Goyal (2026) reported. Section 3.3 shows that the rewriter's capability is exactly the factor separating them: once it is fixed at the level of the model being improved, the gain from automatic rewriting disappears.

What generalises. Of the three configuration-level claims, only one survives every replication we ran. The harness side's inability to absorb budget holds on two task families and at two model scales, and its sign is negative in the second environment. The weight side's gain holds on retrieval and points the same way elsewhere without reaching significance. The advantage of changing both sides holds only on retrieval. A reader deciding where to spend should weight these three differently, and a paper reporting only the first environment would have overstated the third.

What the harness side is buying. The accepted proposals read like competent advice, and roughly half of the harness side's runs still end with the initial harness as the best member of the pool. Taken with the flat curve, this suggests the ceiling is not the quality of the advice but how little the model gains from following it. That is a different target from the one automatic rewriting optimises, and it is where we would look next.

6 Conclusion

Under matched compute, rewriting the harness does not convert budget into success. That null holds on two task families and at two model scales, and in the second environment the trend is negative. Training the weights does convert budget into success where a search returns plenty of material. Changing both is the strongest configuration on the retrieval task, where it alone is significant under every condition, but that advantage does not replicate in the second environment and we do not claim it generally. The gap is not a shortfall in the rewriter: replacing it with a model nine times larger still yields no significant gain. A direct exchange experiment shows further that rewritten harness text transplanted onto trained weights carries almost no gain, so the division of labour in which declarative knowledge goes to the prompt and procedural knowledge to the weights does not hold in this setting. The question "weights or words" has an answer, and it is conditional on the budget and on how much material a search returns.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, including drafting and revising prose in the main text and the appendix, and for code assistance, including writing and refactoring the experiment driver, the accounting module and the plotting scripts. We did not use generative AI tools to generate research ideas, to design the experimental protocol, or to produce any of the reported numbers: every value in the paper is computed by the released code from the released run records. We have reviewed all AI-assisted work. AI-assisted code was verified by rerunning the affected analyses and by checking each reported number against the run record it is derived from; AI-assisted prose was read and edited by the authors, and a mechanical check confirms that no numerical value in the text differs from its source. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work trains and evaluates retrieval agents on HotpotQA and on TextWorldExpress, both public research datasets containing no personal or sensitive data, and involves no human subjects. The models adapted here are open-weight and are used within their licences. The study is a measurement of where a fixed compute budget is best spent and does not propose a capability whose misuse we can foresee; its practical effect, if any, is to discourage spending compute on a method that does not repay it. We report every run behind every table, including those whose result is null or negative, so that a reader can check any claim in the paper against the runs behind it.

REPRODUCIBILITY STATEMENT

The accounting that defines the comparison is given in Section 3.2 and in Equations 1 and 2; the exchange experiment's two estimands are defined in Equations 3 and 4. Hyperparameters common to every run are listed in Appendix A. Every cell of every table is given at seed level in Appendices B, C and D, together with the number of updates each budget bought and the fraction of the allotted budget actually spent, so that a reader can recompute each mean and standard deviation in the main text. The twenty swap-cost observations behind Section 4.4 are listed individually in Appendix E. The run records, the preregistration document with its falsification conditions, and the code that produced every figure and table are released with the paper.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Xiaofan Bai, Hongqiang Lin, Chao Liu, Yantao Zhang, Xuan Jin, Xipeng Cao, and Yuhong Li. Skillzip: Evaluation-free skill compression for self-evolving agents by discovering reusable structure. *arXiv preprint arXiv:2608.11079*, 2026.

Yonatan Gideoni, Sebastian Risi, and Yarin Gal. Simple baselines are competitive with code evolution. *arXiv preprint arXiv:2602.16805*, 2026.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 34133–34156, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in neural information processing systems*, 35:30016–30030, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, et al. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024.

Vivek Kulkarni, Sudipta Paul, Aounon Kumar, Nicholas Tzou, and Srinivas Chappidi. Sbco: Self-supervised, verifier-grounded harness optimization for planning agents. *arXiv preprint arXiv:2608.10157*, 2026.

Yaxin Luo, Haobin Jiang, Jialv Zou, Xu Huang, Wenhao Yan, Haodong Li, Zhengrong Yue, Jing Li, Xiaofu Chen, Xiaohan Zhao, et al. Autodesign: Meta-harness optimization for long-horizon agentic design. *arXiv preprint arXiv:2608.13560*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7957–7968, 2023.

Jaideep Ray and Ankit Goyal. From agent failures to text policies: What works and what breaks. *arXiv preprint arXiv:2607.20668*, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Huatong Song, Fei Bai, Ming Yang, Renyuan Li, Jia Deng, Jujie He, Zhange Zhang, Daixuan Cheng, Yan Xing, Qi Yun, et al. Clawgym ii: Exploring black-box rl on agent harness. *arXiv preprint arXiv:2608.16798*, 2026.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. Trace: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026.

Peidong Wang, Zhiming Ma, Ying Chang, Xufang Luo, Xiaocui Yang, Shi Feng, Yuqing Yang, and Dongsheng Li. Self-evolving embodied agents via skill-harness evolution. *arXiv preprint arXiv:2608.11350*, 2026a.

Xucong Wang, Zhe Zhao, Liheng Yu, Di Wu, Xiaofeng Cao, and Pengkun Wang. Didpo: Diff-in-diff policy optimization for coding agent training. *arXiv preprint arXiv:2608.07147*, 2026b.

Zechuan Wang, Siyuan Lu, Hongxuan Zhang, Linjian Mo, Chenyi Zhuang, and Leilei Gan. Teach the magnitude, not the direction: Verifier-bounded credit assignment for multi-turn multi-step llm agents. *arXiv preprint arXiv:2608.13179*, 2026c.

Tianxin Wei, Zhan Shi, Minhua Lin, Bing He, Zewen Liu, Yisi Sang, Yuanchen Bei, Xuying Ning, Jiaru Zou, Ting-Wei Li, et al. Evo-harness: Context-to-harness skill compilation for self-evolving agents. *arXiv preprint arXiv:2608.15071*, 2026.

Haoze Wu, Chuqiao Kuang, Tianyi Zhuang, and Xiaoguang Li. Beyond outcome rewards: Step-level self-distilled policy optimization for deep search agents. *arXiv preprint arXiv:2608.12764*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

Siqi Yang, Qianlan Yang, Yu-Xiong Wang, Saurabh Pujar, and Martin Hirzel. One recipe, many harnesses: What self-evolution encodes across languages and models. *arXiv preprint arXiv:2608.10178*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Huan Zhang, Mingju Chen, Dongxu Zhou, Can Lv, Heng Chang, Sen Cui, Faguo Wu, and Shiji Zhou. Trca: Transition-wise rubric credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2608.16156*, 2026a.

Lunjun Zhang, Ryan Chen, and Bradly C Stadie. Evolutionary system prompt learning for reinforcement learning in llms. *arXiv preprint arXiv:2602.14697*, 2026b.

Table 4: Main matrix at seed level. Success for seeds 0, 1 and 2; GRPO updates for the same three seeds; and spent FLOPs as a fraction of the budget allotted.

adapted	k	budget	seed 0	seed 1	seed 2	updates	spent / allotted
weights	3	1x	0.528	0.544	0.532	24/24/23	1.07 / 1.06 / 1.06
weights	3	10x	0.650	0.636	0.648	124/123/118	1.03 / 1.02 / 0.97
harness	3	1x	0.524	0.504	0.550	0/0/0	1.03 / 1.02 / 1.01
harness	3	10x	0.544	0.556	0.488	0/0/0	1.00 / 1.00 / 1.00
both	3	1x	0.484	0.528	0.520	11/21/11	0.91 / 0.97 / 0.92
both	3	10x	0.648	0.642	0.604	84/82/85	0.52 / 0.95 / 0.98
weights	2	1x	0.544	0.520	0.534	4/4/4	1.18 / 1.11 / 1.13
weights	2	10x	0.624	0.632	0.604	62/66/65	0.93 / 0.98 / 0.97
harness	2	1x	0.502	0.534	0.554	0/0/0	1.22 / 1.09 / 1.01
harness	2	10x	0.504	0.564	0.524	0/0/0	1.01 / 1.00 / 1.00
both	2	1x	0.514	0.520	0.522	5/6/6	0.89 / 0.92 / 0.94
both	2	10x	0.574	0.584	0.600	46/41/26	0.72 / 0.94 / 0.64
weights	1	1x	0.368	0.364	0.374	1/1/1	0.84 / 0.82 / 0.86
weights	1	10x	0.348	0.398	0.434	17/23/23	0.67 / 0.92 / 0.93
harness	1	1x	0.374	0.358	0.368	0/0/0	1.02 / 1.11 / 1.10
harness	1	10x	0.366	0.404	0.380	0/0/0	1.01 / 1.01 / 1.01
both	1	1x	0.332	0.390	0.346	2/2/2	0.76 / 0.84 / 0.78
both	1	10x	0.428	0.402	0.434	23/16/18	0.94 / 0.59 / 0.90

A HYPERPARAMETERS

All runs share the settings below; only the budget, the retrieval condition and the seed vary across cells. LoRA is applied to the attention and MLP projections with rank 32. Each GRPO step draws 16 prompts and samples a group of 8 completions per prompt, with a minibatch of 16 for the update. The rewriting side keeps a Pareto pool of 48 candidates and runs one generation of rewriting every 5 GRPO steps in the alternating configuration. Validation runs at every quarter of the budget, and the swap probe of Section 4.4 evaluates 32 held-out items before and after each swap. Episodes are capped at 12 turns, and the training reward is the same strict span scorer used for the reported success rate.

B PER-SEED RESULTS

Table 4 gives every cell of the main matrix at seed level, together with the number of GRPO updates the budget bought and the fraction of the allotted budget actually spent. The means and standard deviations in Table 1 are computed from the three success values in each row.

Two patterns in the last column are worth separating from the success figures. Rewriting the harness never underspends, because its cost is dominated by evaluating candidates on a fixed pool and the lookahead gate rarely has to stop it early. The alternating configuration underspends the most, and most severely at the high budget, because a GRPO step late in training is expensive enough that the gate refuses to start one. The weight-only configuration falls between the two.

C REPLICATION AT 9B AND ON THE SECOND ENVIRONMENT

Table 5 gives the 9B replication at seed level and Table 6 the second environment; the means and intervals quoted in Sections 4.3 and 4.8 are computed from these values.

D CONTROLS

Table 7 gives the three control probes at seed level. Their means are the reference values quoted in the caption of Table 1.

Table 5: Cross-scale replication at 9B, seed level, at the plentiful-material condition.

	1x			10x		
adapted	seed 0	seed 1	seed 2	seed 0	seed 1	seed 2
weights	0.498	0.516	0.508	0.616	0.594	0.592
harness	0.610	0.570	0.618	0.570	0.608	0.546

Table 6: Second environment, seed level.

	1x			10x		
adapted	seed 0	seed 1	seed 2	seed 0	seed 1	seed 2
weights	0.412	0.360	0.366	0.406	0.432	0.410
harness	0.366	0.330	0.378	0.378	0.260	0.276
both	0.400	0.406	0.364	0.398	0.404	0.400

Table 7: Control probes at seed level. The unchanged agent and the hand-written harness are evaluated under each retrieval condition; the retrieval-forbidden probe answers from parametric memory and is therefore independent of k.

control	k	seed 0	seed 1	seed 2	mean
unchanged	3	0.550	0.510	0.544	0.535
unchanged	2	0.536	0.518	0.562	0.539
unchanged	1	0.372	0.364	0.342	0.359
hand-written	3	0.518	0.506	0.504	0.509
hand-written	2	0.482	0.482	0.484	0.483
hand-written	1	0.318	0.306	0.300	0.308
retrieval-forbidden	n/a	0.050	0.046	0.062	0.053

E SWAP-COST OBSERVATIONS

Table 8 lists every swap probe behind the estimate in Section 4.4. Each row records the generation at which the harness was swapped, the accuracy of the current weights on the held-out probe set immediately before and immediately after the swap, and their difference. A positive difference is a loss. Cells of the alternating configuration that completed no generation contribute no rows, which is why the twenty observations are unevenly distributed across conditions.

Table 8: All swap-cost observations.

run	generation	before	after	loss
fC_m0_hi_s1	3	0.5938	0.6250	−0.0312
fC_m0_hi_s1	8	0.6562	0.5625	+0.0938
fC_m0_hi_s1	9	0.5000	0.5312	−0.0312
fC_m0_hi_s1	10	0.6562	0.6562	+0.0000
fC_m0_hi_s2	3	0.5312	0.5000	+0.0312
fC_m0_hi_s2	7	0.5938	0.5625	+0.0312
fC_m0_hi_s2	10	0.5312	0.6250	−0.0938
fC_m0_hi_s2	11	0.6250	0.5312	+0.0938
fC_m0_hi_s2	15	0.6250	0.5312	+0.0938
fC_m0_lo_s0	1	0.6250	0.5000	+0.1250
fC_m0_lo_s1	4	0.5312	0.4688	+0.0625
fC_m0_lo_s2	1	0.5000	0.3750	+0.1250
fC_m0_lo_s2	2	0.5000	0.5000	+0.0000
fC_m3_hi_s0	8	0.5312	0.5312	+0.0000
fC_m3_hi_s1	2	0.6250	0.4688	+0.1562
fC_m3_hi_s1	3	0.5312	0.5625	−0.0312
fC_m3_hi_s2	1	0.5625	0.5312	+0.0312
fC_m3_lo_s0	1	0.3750	0.3750	+0.0000
fC_m6_hi_s0	4	0.4062	0.4062	+0.0000
fC_m6_hi_s2	1	0.2812	0.3125	−0.0312