AGENTTIME: BENCHMARKING TIME AWARENESS OF LLM AGENTS IN LONG HORIZON TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tasks require large language model (LLM) agents to allocate time, adapt execution, and deliver useful results within the time budgets. Existing evaluations provide limited evidence connecting these decisions to task quality and timely delivery across professional workflows. We introduce AgentTime, a benchmark comprising 82 long-horizon tasks from six benchmarks across nine professional domains. Three scenarios cover ample budgets, tight budgets, and unexpected budget reductions. Using plans, execution traces, and final scores, AgentTime assesses five dimensions: punctuality, subtask time estimation, time allocation and scheduling, time-adaptive execution, and time-use efficiency. Preliminary aggregate results across four agent–instruction configurations from three systems yield three findings. Larger fixed budgets improve task quality for all four configurations, with diminishing gains per additional minute for three. When a 60-minute budget is shortened to 45 minutes during execution, all four configurations score above their separate fixed 30-minute baselines; this comparison does not isolate the effect of the reduction. Explicit time-management instructions yield mixed results in the Codex case study: task score improves by 20.9% at 30 minutes and also improves under budget reduction, but not at 10 or 120 minutes; planning and execution scores do not improve uniformly. AgentTime provides a common framework for diagnosing time-management strengths and limitations in long-horizon agents.

1 INTRODUCTION

Long-horizon tasks require LLM agents to manage complex workflows, from web interaction to research engineering (Liu et al., 2024; Zhou et al., 2024; Mialon et al., 2024; Wijk et al., 2024). Delegating these workflows also delegates decisions about time spent on exploration, execution, and verification, and when to deliver. Under a time budget, additional effort on one subtask reduces the time available for other essential work. Agents must therefore align the scope, order, and depth of their work with the available time to deliver useful outcomes on time.

We define *time awareness in long-horizon tasks* as the ability to track the time available and adjust planning and execution to achieve task goals. This operational definition draws on time estimation and self-regulation in human time management (Brady et al., 2022), and on time allocation and execution monitoring in anytime computation (Zilberstein, 1996). Agents must estimate subtask durations, allocate time by priority, and revise their remaining work as execution unfolds or budgets change. Figure 1 illustrates these decisions: faster extraction and fact checking before polishing enable reports on all 70 documents within ten minutes.

Evaluating time awareness requires connecting an agent's time-related decisions throughout a task with the quality and timeliness of its final output. Benchmarks for reasoning about time test understanding of time information (Tan et al., 2023; Chu et al., 2024), while TimeArena and Robotouille assess scheduling with simulated durations (Zhang et al., 2024; Gonzalez-Pumariega et al., 2025). RE-Bench and negotiation studies examine execution within time budgets (Wijk et al., 2024; Sehgal et al., 2026), and BATS and Timely Machine develop budget-aware planning and execution methods (Liu et al., 2026b; Ma et al., 2026). Although these efforts include process analyses, a benchmark spanning diverse long-horizon tasks is still needed to jointly assess subtask time estimation, time

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

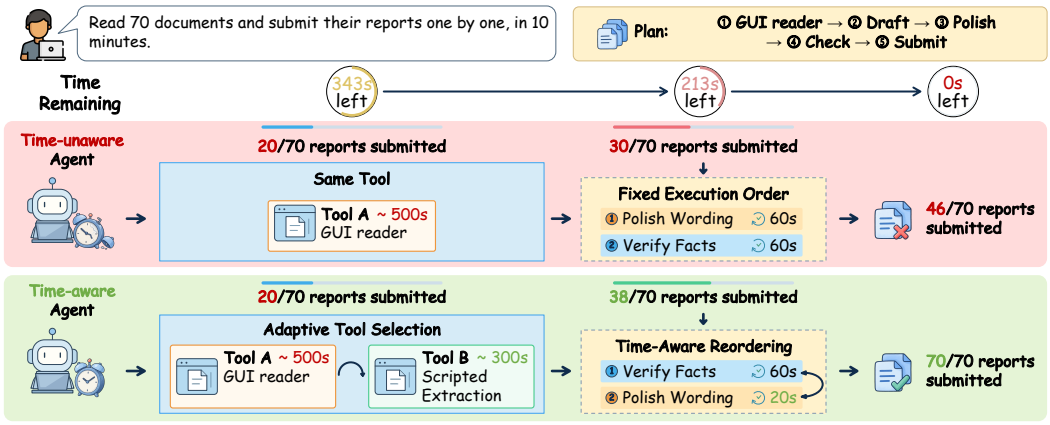


Figure 1: Comparison between agents with good and poor time awareness.

allocation and scheduling, and time-adaptive execution alongside timely delivery and quality gains across time budgets.

In this paper, we introduce **AgentTime**, a benchmark that connects agents' planning and execution with the quality and timeliness of their outputs on long-horizon tasks. Its evaluation framework uses initial plans, timestamped execution traces, and final task scores to assess five dimensions: punctuality, subtask time estimation, time allocation and scheduling, time-adaptive execution, and time-use efficiency. This structure supports interpreting final task performance in light of the agent's time estimates, priorities, and adjustments during execution.

To create meaningful demands on time management, we curate 82 tasks across nine professional domains from six existing benchmarks, including Agents' Last Exam and Long-Horizon Terminal-Bench (Sun et al., 2026; Li et al., 2026b). We select long-horizon workflows that admit multiple solution paths, support graded progress, and allow further refinement. These properties create meaningful choices about time use and allow partial progress to be evaluated. Three scenarios vary budget tightness and stability: ample budgets permit refinement, tight budgets require prioritization, and reductions during execution test timely delivery under revised constraints.

The available aggregate results across four agent–instruction configurations yield three findings (Section 4.2). First, larger fixed budgets improve task quality for all four configurations, but gains per additional minute diminish for three; Codex gains more per minute from 30 to 120 minutes than from 10 to 30 minutes. Second, when a 60-minute budget is shortened to 45 minutes at minute 30, all four configurations score above their separate fixed 30-minute baselines. Without fixed 45-minute and unchanged 60-minute controls, this comparison cannot isolate the effect of the reduction. Third, explicit time-management guidance has mixed effects in the Codex case study: task scores improve at 30 minutes and under budget reduction, but not at 10 or 120 minutes, while allocation and execution-adjustment scores do not improve uniformly.

The paper makes three contributions:

- We introduce an evaluation framework for time awareness that connects subtask time estimates, time allocation and scheduling, and execution adjustments with submission timing and gains in task quality across budgets.
- We curate 82 long-horizon tasks across nine professional domains and design three scenarios that vary budget tightness and stability. Task selection emphasizes multiple solution paths and graded progress, enabling evaluation of how agents prioritize and refine their work under time constraints.
- We characterize quality gains across fixed budgets, compare performance under an unexpected budget reduction with fixed-budget baselines, and examine the mixed effects of explicit time-management guidance in a controlled Codex prompt comparison.

Table 1: Five dimensions of time awareness in AgentTime.

ID	Dimension	Description
A1	Punctuality	Assess final submission timing relative to the time budget, accounting for late, missing, or invalid submissions and interpreted alongside task quality.
A2	Subtask Time Estimation	Estimate the time needed for each subtask, including reasoning, tool use, testing, and retries, allowing for uncertainty.
A3	Time Allocation and Scheduling	Distribute available time across subtasks, tool use, agent-run testing, and finalization according to task priorities, dependencies, and estimated subtask times.
A4	Time-adaptive Execution	Adjust subtask selection, order, duration, or tool-use methods when the remaining time budget no longer matches the current plan.
A5	Time-Use Efficiency	Assess gains in final task scores across time budgets relative to the room for improvement over the baseline score.

2 Related Work

We review existing work on time awareness in LLM agents, covering both evaluation benchmarks and methods for acting under time and resource budgets.

2.1 Benchmarking Time Awareness in Language Models and Agents

Temporal reasoning benchmarks treat time as information to interpret: TempReason and TimeBench test reasoning about time-dependent facts, event order, and durations (Tan et al., 2023; Chu et al., 2024). Agent evaluations examine how time information affects behavior. Discrete Minds examines responses to elapsed time and urgency, TicToc tests information refreshing, and a self-duration study probes agents' estimates of their own execution time (Wang et al., 2025; Cheng et al., 2026; Garikaparthi, 2026). TimeArena and Robotouille evaluate scheduling with action durations and dependencies, while Gaia2 evaluates behavior in dynamic, asynchronous environments (Zhang et al., 2024; Gonzalez-Pumariega et al., 2025; Froger et al., 2026). However, these evaluations provide limited evidence on how agents choose the scope and depth of professional work when further refinement competes with timely delivery.

Related evaluations connect resource constraints with task outcomes and agent behavior. RE-Bench compares research-engineering performance across time budgets, while Agents' Last Exam assesses long-horizon professional workflows (Wijk et al., 2024; Sun et al., 2026). Real-Time Deadlines analyzes negotiation behavior and timely agreement under remaining-time feedback (Sehgal et al., 2026). Timely-Eval examines performance and strategy adaptation across time budgets and tool latencies (Ma et al., 2026). BAGEN assesses remaining-resource estimates, task feasibility, and early failure detection through trajectory replay (Lin et al., 2026). However, despite including process analyses, these evaluations do not jointly assess the accuracy of prospective subtask time estimates, the alignment of time allocations with task priorities, and implemented adjustments when plans no longer fit the remaining time. AgentTime evaluates these decisions together and relates them to output quality and timely delivery across professional tasks. Appendix D compares evaluation coverage along A1–A5.

2.2 Agents under Time and Resource Budgets

Anytime computation studies the tradeoff between solution quality and computation time (Zilberstein, 1996). Recent methods expose resource budgets to language models: BATS adapts planning and verification to remaining tool calls, and BudgetThinker controls reasoning through remaining-token signals (Liu et al., 2026b; Wen et al., 2025). These methods make resource consumption explicit. However, token and tool-call budgets do not directly determine elapsed execution time: model latency, tool execution, and retries vary in duration. Adherence to these budgets therefore does not by itself ensure completion by a deadline.

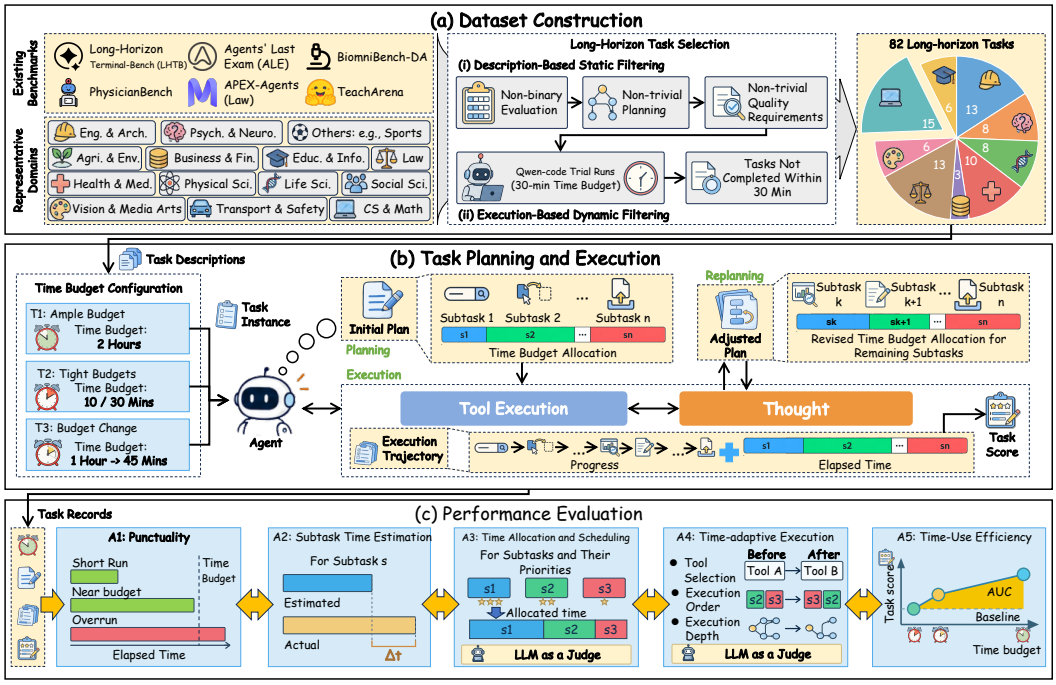


Figure 2: AgentTime overview: (A) dataset construction, (B) task planning and execution, and (C) performance evaluation.

Timely-RL trains agents to adapt reasoning and tool use to temporal feedback (Ma et al., 2026). Engagement Process separates action and observation streams to represent deliberation latency, persistent actions, and delayed feedback (Li et al., 2026a). These approaches make time explicit in training and interaction design. Assessing their effectiveness in long-horizon professional workflows requires linking changes in planning and execution to output quality and submission timing. AgentTime provides a common protocol for examining these relationships across tasks and time budgets.

3 AgentTime

AgentTime benchmarks LLM agents' time awareness in long-horizon tasks across diverse professional domains. We first describe dataset construction, then introduce three evaluation scenarios that vary time-budget tightness and stability. Finally, we define the five time-awareness dimensions in Table 1, assessed using plans, execution traces, and final task scores.

3.1 Dataset Construction

We select complex, long-horizon tasks to observe how agents adjust plans and actions as time is consumed or budgets change. Diverse tools, task dependencies, and deliverables create different demands on time estimation, allocation, and execution adjustments. Following the domain taxonomy of Agents' Last Exam (ALE), we build a candidate collection from six existing benchmarks: Long-Horizon Terminal-Bench (LHTB), ALE, BiomniBench-DA, PhysicianBench, APEX-Agents (law), and TeachArena (Li et al., 2026b; Sun et al., 2026; Qu et al., 2026; Liu et al., 2026a; Vidgen et al., 2026; TeachArena authors, 2026). The resulting dataset contains 82 tasks spanning nine of ALE's thirteen domains; Table 4 in Appendix A.1 reports the domain distribution by source.

Screening combines static analysis of task descriptions and scoring criteria with Qwen-code (Qwen-3.8-flash) trial runs to retain tasks with meaningful time-allocation choices and measurable progress. Tasks must support non-binary scoring so that partial progress under different time budgets can be distinguished. They must admit multiple solution paths, allowing agents to adjust their approach,

and permit continued improvement, such as strengthening output robustness, so that additional time can yield higher-quality results. Finally, we retain tasks that Qwen-code does not complete within a 30-minute screening budget, providing sustained execution over which time estimation, allocation, and adjustments can be examined. Appendix A.1 details the screening records and task-level audit requirements.

The dataset includes both non-continuation tasks and continuation tasks that allow intermediate submissions for verification. Appendix A.1 describes these submission modes and the controller that excludes evaluator verification time from the agent's time budget.

3.2 Three Evaluation Scenarios

Using the constructed dataset, we design three evaluation scenarios to simulate real-world agents executing long-horizon tasks under different time budgets and settings (Figure 2, middle).

T1: Ample Budget. Ample time allows agents to carry out long-horizon tasks thoroughly and refine their outputs. Therefore, we provide agents with a 120-minute budget to execute each task.

T2: Tight Budgets. Tight time budgets require agents to prioritize essential work while maintaining task performance. Therefore, we run agents on every task separately with budgets of 10 and 30 minutes.

T3: Budget Change. An unexpected reduction in available time requires agents to adjust their ongoing work and bring the task to a timely close. Therefore, we initially assign a 60-minute budget and, at minute 30, halve the remaining time from 30 to 15 minutes, giving a revised total budget T of

$$T = 30 + \frac{60 - 30}{2} = 45 \text{ minutes}. \tag{1}$$

We select 120 minutes as an ample budget and 10 and 30 minutes as tight budgets based on task execution-time statistics and screening results from dataset construction. Appendix A.4 gives the execution protocol, and Appendix C.1 provides the agent instructions.

3.3 Time Awareness Dimensions

Using the plans, execution traces, and final scores from these scenarios, we evaluate five dimensions of time awareness. We represent a task run by $\mathcal{R} = (P, E, T, S)$, where P is the initial task plan, E is the full timestamped execution trace, T is the total scoring budget after any revision, and $S \in [0, 1]$ is the final task score. All durations use charged elapsed time, which excludes controller-authorized evaluator verification but includes planning, inference, ordinary tools, and agent-run tests (Appendix A.3).

For each of the n subtasks s_i in P, indexed by $i = 1, \ldots, n$, let t_i and t'_i denote its planned time allocation and actual execution time, respectively. The plan's allocations and the subtask-time summary E_{sub} extracted from E are

$$P = \{(s_i, t_i)\}_{i=1}^{n}, \qquad E_{\text{sub}} = \{(s_i, t'_i)\}_{i=1}^{n}. \tag{2}$$

We align subtasks in E with those in P, assigning zero execution time to unexecuted planned subtasks and merging unmatched executed subtasks into the preceding subtask in the trace. The full trace E also retains action order, tool feedback, completion status, and plan revisions. For A1, the aligned times cover all charged time from run start to final submission, including initial planning, with overlapping execution counted only once.

A1: Punctuality. Drawing on prior work on human punctuality (Francis-Smythe & Robertson, 1999), we define A1 to measure the timing of a valid final submission relative to the time budget:

$$A1 = \begin{cases} \dfrac{\sum_{i=1}^{n} t'_i}{T}, & \sum_{i=1}^{n} t'_i \le T \text{ and } S \neq 0, \\ 0, & \sum_{i=1}^{n} t'_i > T, \\ 1, & \sum_{i=1}^{n} t'_i \le T \text{ and } S = 0. \end{cases} \tag{3}$$

A1 equals the fraction of the time budget used for a valid on-time submission with a nonzero score, zero for a late submission, and one for a valid on-time submission with a zero score. Missing or invalid submissions also receive zero. We interpret A1 alongside S; it has no preferred direction.

A2: Subtask Time Estimation. Prior work finds that LLMs struggle to estimate their own execution time, including in multi-step agentic settings (Garikaparthi, 2026). A2 evaluates this capability at the planned-subtask level.

For subtask s_i, let $\widehat{t_i}$ be its time estimate recorded in plan P before execution. Let $\mathcal{I}_2$ contain the indices of completed subtasks with recorded estimates and complete timing records.

A2 sums their estimation errors, normalized by the total budget T:

$$A2 = \sum_{i \in \mathcal{I}_2} \left| \frac{t'_i - \widehat{t_i}}{T} \right|. \tag{4}$$

Lower A2 values indicate more accurate time estimates. A2 is unavailable when $\mathcal{I}_2$ is empty. The reporting protocol records the proportion of subtasks included and the reasons for exclusions (Appendix A.5).

A3: Time Allocation and Scheduling. Research on human time management links a focus on task importance to greater time investment in important tasks (Yuan & Sun, 2025). Following this intuition, A3 uses LLM judges to assess whether time allocations in P reflect subtask priorities and contributions to the overall goal. Higher-priority subtasks should generally receive relatively more time, while accounting for their time requirements and dependencies. Let $\mathrm{LLM}(\cdot) \in [0, 1]$ denote the calibrated expected rating, normalized as described below. Given P and T, when all n subtasks can be rated, we define

$$A3 = \frac{1}{n} \sum_{i=1}^{n} \mathrm{LLM}(s_i, t_i \mid P, T). \tag{5}$$

Higher $A3 \in [0, 1]$ indicates allocations better aligned with task priorities. Subtasks with insufficient evidence are marked *Unable to Determine* (UTD); we average only the rated subtasks. The reporting protocol requires their proportion and UTD counts. A3 is unavailable if none can be rated. Judgments use only information available at planning time. Appendix C.2 gives the detailed LLM scoring rubric.

A4: Time-adaptive Execution. Time budgets may change during execution, and actual subtask durations may differ from planned allocations. Drawing on research on human time regulation (Liu & Li, 2026), A4 assesses whether the agent adapts subtask selection, order, duration, or tool-use methods to task priorities and the remaining budget. We compute

$$A4 = \mathrm{LLM}(E \mid P, T). \tag{6}$$

Higher $A4 \in [0, 1]$ indicates better-supported time-adaptive execution. Runs with no observed budget-plan mismatch or insufficient evidence receive UTD; the reporting protocol requires coverage and UTD counts. Appendix C.3 gives the prompt.

The protocol uses three different LLM judges for both A3 and A4. Judges $j \in \{1, 2, 3\}$ independently rate each subtask allocation (A3) or full execution trace (A4) on a four-point scale $\{0, 1, 2, 3\}$.

To adjust for scoring strictness, the protocol jointly calibrates their ratings using a many-facet Rasch model (Linacre et al., 1990). The calibrated expected rating, divided by three, defines $\mathrm{LLM}(\cdot)$. Calibration is fitted separately for each dimension and budget condition; Appendix A.6 gives the details.

A5: Time-Use Efficiency. For the same task, an agent should adapt the depth of its work to the available time. A short budget calls for a usable result quickly, while a longer budget allows more thorough execution, comprehensive checking, and reflection on the result. A5 evaluates how effectively agents use additional time to improve task performance across different budgets.

Consider matched runs with the same task instance, agent configuration, and repetition index, at budgets ranging from a short baseline T_0 to the maximum total budget $T_{\max}$. Let b denote the time-budget coordinate, $S(b)$ the linear interpolation of final task scores across these runs, and $S_{T_0} = S(T_0)$ the baseline score. We measure improvement beyond T_0 using a headroom-normalized area under the curve (AUC):

$$A5 = \frac{1}{T_{\max}} \int_{T_0}^{T_{\max}} \frac{S(b) - S_{T_0}}{1 - S_{T_0}}\, db. \tag{7}$$

Following the normalized-gain principle (Hake, 1998), dividing by $1 - S_{T_0}$ expresses improvement as a fraction of the remaining room for improvement, accounting for differences in agents' baseline task performance. The factor $1/T_{\max}$ normalizes the area by the maximum total budget.

We retain negative gains and observed failure scores. A baseline of one or a missing input makes A5 unavailable. Each curve combines final scores from separate runs; in our evaluation, it also includes a budget-change condition (Section 3.2). Appendix A.5 gives the interpolation and eligibility rules.

The five dimensions are reported separately. The aggregation protocol averages repetitions within instances, instances within tasks, and then tasks equally, computing A5 within each matched group first. It requires rating distributions, missing-evidence counts, and uncertainty clustered by task or shared template (Appendix A.7). The available aggregates do not yet supply all these records; Appendix B summarizes the outstanding evidence.

4 Experiments

We select state-of-the-art LLM agents to perform the long-horizon tasks in AgentTime under the three evaluation scenarios defined in Section 3.2. We evaluate the agents along our five time-awareness dimensions (A1–A5) and report their final task scores.

4.1 Experimental Setup

Agent configurations. We compare three state-of-the-art large language model agents: **Codex (GPT-5.6-sol)**, **Qwen-code (Qwen-3.8-flash)**, and **DeepSeek Harness (DeepSeek-v4-pro)**. To reduce the influence of network conditions, we access all models through their APIs and deploy the agent environments on a local server. The local-environment record reports an average API throughput of 42.32 tokens/s.

Task instructions. For long-horizon tasks, the instructions state the time budget in seconds and ask the agent to submit within the time budget. Before starting execution, the agent must create a plan specifying the time allocated to each subtask. The instructions provide `python .ale/time_budget.py` to inspect the current time, elapsed time, and remaining budget.

To examine how much the observed time awareness depends on explicit instruction following, we compare two prompt variants for Codex (GPT-5.6-sol). The **Simple prompt** follows the requirements above. The **Instructed prompt** additionally states that exceeding the time budget causes task failure, requires periodic time checks, and instructs the agent to adjust its remaining plan and execution strategy when actual progress differs from the plan. Appendices C.1.1 and C.1.2 provide the complete templates.

We therefore evaluate four configurations: **Codex** and **Codex (Instructed)** (both GPT-5.6-sol), **Qwen-code** (Qwen-3.8-flash), and **DeepSeek Harness** (DeepSeek-v4-pro). Codex (Instructed) uses the Instructed prompt; all other configurations use the Simple prompt.

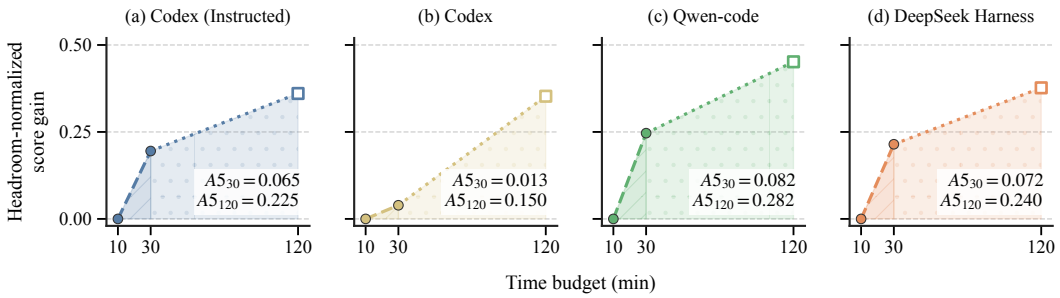


Figure 3: Time-use efficiency from Table 2's mean task scores. Shading shows headroom-normalized gains; $A5_T$ divides the area from 10 to T minutes by T. These aggregate curves connect separate fixed-budget runs and exclude T3.

Table 2: Fixed-budget results for four agent–instruction configurations.

Setting	Agent configuration	$S\uparrow$	A1	A2↓	A3↑	A4↑
T2: 10 min	Codex	0.3907	0.7390	0.5577	0.6564	0.7567
	Qwen-code	0.1436	0.1217	1.2532	0.2720	0.1751
	DeepSeek Harness	0.2111	0.1637	1.0934	0.2441	0.2374
	Codex (Instructed)	0.3804	0.6132	–	0.6271	0.7531
T2: 30 min	Codex	0.4146	0.7507	0.7052	0.6493	0.6364
	Qwen-code	0.3547	0.4573	1.2898	0.2570	0.5654
	DeepSeek Harness	0.3804	0.4272	0.9319	0.3582	0.4510
	Codex (Instructed)	0.5013	0.6798	–	0.6656	0.7005
T1: 120 min	Codex	0.6056	–	0.5761	0.7388	0.6311
	Qwen-code	0.5304	–	0.6797	0.3645	0.7413
	DeepSeek Harness	0.5084	–	0.8597	0.3488	0.5680
	Codex (Instructed)	0.6038	–	–	0.6636	0.6593

4.2 Result Analysis

We examine agents' time awareness across fixed time budgets, under a time-budget reduction during execution, and with more explicit time-management instructions. Budget comparisons hold each agent–instruction configuration fixed. System comparisons under the Simple prompt include differences in both the backbone and harness; the instruction comparison uses Codex as a case study, holding these fixed while varying the prompt.

Finding 1: Larger time budgets improve task quality, with diminishing gains for three of four configurations. All four configurations improve in task quality as the time budget increases (Table 2). For Qwen-code, DeepSeek Harness, and Codex (Instructed), extending the budget from 10 to 30 minutes raises S by 0.1209–0.2111, while extending it from 30 to 120 minutes adds 0.1025–0.1757. In these three cases, the additional 90 minutes yield smaller gains than the first additional 20 minutes. Codex instead gains 0.0239 and 0.1910, reaching $S = 0.6056$ at 120 minutes; its gain per added minute increases by 77.6% over the later interval.

These comparisons use final scores from separate runs, not progress within a run. Higher quality does not establish uniformly stronger time awareness: A1 has no preferred direction, A2 depends on budget normalization and eligible subtask coverage, and A3/A4 are calibrated separately by budget. Under the ample 120-minute budget, we assume that agents submit a final result and do not compute A1; these entries are shown as dashes. Codex (Instructed)'s A2 entries are also dashes: explicit replanning changes its plan during execution, so actual subtask durations are no longer directly comparable to the initial plan's time estimates. Figure 3 summarizes fixed-budget mean scores, with aggregate $A5_{120}$ values of 0.1502–0.2823. These areas exclude T3 and do not replace the matched-group A5 defined in Section 3.3; Appendix B explains the distinction.

Table 3: T3 results after the budget shrinks from 60 to 45 minutes at minute 30.

Agent configuration	$S \uparrow$	A1
Codex	0.4455	0.6617
Qwen-code	0.4941	0.6639
DeepSeek Harness	0.4665	0.5444
Codex (Instructed)	0.5433	0.6770

Design insight. This pattern motivates allocating time according to expected quality gain per unit time: secure a usable result early, then select refinements that fit the remaining budget.

Finding 2: Task quality under a shortened budget exceeds the fixed 30-minute baseline. T3 halves the remaining budget at minute 30, giving a revised total of 45 minutes. Table 3 reports the resulting task scores and A1.

All four configurations score 0.0309–0.1394 higher than in separate fixed 30-minute runs. Across all four configurations, T3 scores reach 73.6–93.2% of the respective 120-minute scores (Table 2). A1 has no preferred direction and cannot independently rank punctuality.

Runs completed before the notice are included, so these aggregates do not measure response speed. Missing fixed 45-minute and unchanged 60-minute controls also prevent isolating the effect of budget reduction.

Design insight. Budget changes motivate explicit replanning: reprioritize unfinished subtasks against the revised remaining time and reserve time to validate and submit the current result.

Finding 3: More explicit instructions do not guarantee stronger agent time awareness. With Codex's backbone and harness fixed, Codex (Instructed) yields lower A3/A4 and task quality than Codex at 10 minutes, but higher values at 30 minutes; task quality is also higher under T3 (Tables 2 and 3). At 120 minutes, its task score is slightly lower (0.6038 versus 0.6056), with lower A3 (0.6636 versus 0.7388) but higher A4 (0.6593 versus 0.6311). A3/A4 comparisons share calibration within each budget.

The instruction gains are therefore inconsistent across budgets and dimensions. A2 is unavailable for Codex (Instructed); matched-group A5 comparisons and uncertainty estimates remain unavailable.

Design insight. Future agents could use runtime rules linking clock readings to replanning and stopping, with monitoring effort adapted to the remaining budget. These design directions remain to be tested.

5 Conclusion

In this paper, we propose AgentTime, a benchmark for evaluating time awareness in long-horizon tasks, comprising 82 tasks across nine professional domains and three scenarios covering ample budgets, tight budgets, and unexpected budget reductions. Its five-dimensional framework connects subtask time estimation, time allocation and scheduling, and execution adjustments with timely delivery and task quality across budgets. Preliminary aggregate results across three agent systems and four agent–instruction configurations yield three findings. Larger fixed budgets improve task quality for all four configurations, with diminishing gains per additional minute for three. Under an unexpected reduction from 60 to 45 minutes, all four score above their separate fixed 30-minute baselines, although missing matched controls prevent isolating the effect of the reduction. Explicit time-management guidance has mixed effects across budgets and dimensions in the Codex case study. AgentTime provides a common basis for evaluating time-management decisions alongside task quality and timely delivery.

In future studies, we will evaluate a broader range of LLM agents to examine the generality of these findings. We will also extend the current single-agent setting to assess time awareness during multi-agent collaboration, including how agents coordinate time allocation and adapt to shared deadlines.

AI USE STATEMENT

Generative AI tools, including OpenAI Codex and collaborating writing agents, assisted with manuscript drafting and revision, literature searches, protocol and analysis design, LaTeX editing, and plotting code. They also generated synthetic examples to aid visualization design and checked consistency across sections. These design examples are excluded from the reported benchmark measurements. Human verification of the methodology, references, code, and final claims remains necessary before submission.

ETHICS STATEMENT

AgentTime evaluates agents in sandboxed task environments. Adapted tasks must retain source licenses, access restrictions, and privacy protections, particularly for clinical and professional records. Released assets must be permitted for redistribution, and traces must exclude credentials and personal information. Benchmark performance alone does not establish suitability for autonomous use in clinical, legal, or other consequential settings.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A define the protocol, scores, and required records. The overview-table scripts generate the reported configuration results from the supplied aggregates in `data/reported_results.tex`. The main plotting script reads Table 2's mean task scores and reproduces the fixed-budget normalized-gain curves. Audited task manifests and raw run records remain to be provided, as detailed in Appendix B. Reproducing empirical results will require the frozen task inventory, source versions, agent configurations, graders, and timestamped run records specified in the protocol.

REFERENCES

Anna C. Brady, Christopher A. Wolters, and Shirley L. Yu. Self-regulation of time: The importance of time estimation accuracy. *Frontiers in Psychology*, 13:925812, 2022. doi: 10.3389/fpsyg.2022.925812. URL https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2022.925812/full.

Yifu Cai, Xinyu Li, Mononito Goswami, Michał Wiliński, Gus Welter, and Artur Dubrawski. TimeSeriesGym: A Scalable Benchmark for (Time Series) Machine Learning Engineering Agents, 2025. URL https://arxiv.org/abs/2505.13291.

Yize Cheng, Arshia Soltani Moakhar, Chenrui Fan, Parsa Hosseini, Kazem Faghih, Zahra Sodagar, Wenxiao Wang, and Soheil Feizi. Your LLM Agents are Temporally Blind: The Misalignment Between Tool Use Decisions and Human Time Perception. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37082–37104, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1848. URL https://aclanthology.org/2026.findings-acl.1848/.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. TimeBench: A Comprehensive Evaluation of Temporal Reasoning Abilities in Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1204–1228. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.66. URL https://aclanthology.org/2024.acl-long.66/.

Bahare Fatemi, Seyed Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of Time: A Benchmark for Evaluating LLMs on Temporal Reasoning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 94426–94447, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/eb7295a8bc613b375726659c2ecd6f14-Paper-Conference.pdf.

Jan Francis-Smythe and Ivan Robertson. Time-related individual differences. *Time & Society*, 8(2–3):273–292, 1999. doi: 10.1177/0961463X99008002004. URL https://eprints.worc.ac.uk/277/.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Gaia2: Benchmarking LLM Agents on Dynamic and Asynchronous Environments. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 119758–119789, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c26a67e0470774df98c12480ec5d2d7b-Paper-Conference.pdf.

Aniketh Garikaparthi. Can LLMs perceive time? an empirical investigation. In *I Can't Believe It's Not Better Workshop at ICLR*, 2026. URL https://arxiv.org/abs/2604.00010.

Gonzalo Gonzalez-Pumariega, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. Robotouille: An Asynchronous Planning Benchmark for LLM Agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d033407a1a4b7a75cd5f9e4575ad9fb5-Paper-Conference.pdf.

Richard R. Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 66(1):64–74, 1998. doi: 10.1119/1.18809. URL https://doi.org/10.1119/1.18809.

Jialian Li, Yuchen Cao, Junhong Liu, Weiran Guo, Xutao Wang, Jiaming Song, Jiahao Zhang, and Jie Chen. Engagement Process: Rethinking the Temporal Interface of Action and Observation, 2026a. URL https://arxiv.org/abs/2605.11484.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-Horizon-Terminal-Bench: Testing the Limits of Agents on Long-Horizon Terminal Tasks with Dense Reward-Based Grading. arXiv preprint, 2026b. URL https://arxiv.org/abs/2607.08964.

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, and Manling Li. BAGEN: Are LLM Agents Budget-Aware?, 2026. URL https://arxiv.org/abs/2606.00198.

John M. Linacre, Benjamin D. Wright, and Mary E. Lunz. A facets model for judgmental scoring. MESA Memo 61, MESA, July 1990. URL https://www.rasch.org/memo61.htm.

Fang Liu and Man Li. Development and validation of the college students' time management competence scale. *Frontiers in Psychology*, 17:1885044, 2026. doi: 10.3389/fpsyg.2026.1885044. URL https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2026.1885044/full.

Ruoqi Liu, Imran Q. Mohiuddin, Austin J. Schoeffler, Kavita Renduchintala, Ashwin Nayak, Prasantha L. Vemu, Shivam C. Vedak, Kameron C. Black, John L. Havlik, Isaac Ogunmola, Stephen P. Ma, Roopa Dhatt, and Jonathan H. Chen. PhysicianBench: Evaluating LLM Agents in Real-World EHR Environments. arXiv preprint, 2026a. URL https://arxiv.org/abs/2605.02240.

Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-Aware Tool Use Enables Effective Agent Scaling. In *Third Conference on Language Modeling*, 2026b. URL https://arxiv.org/abs/2511.17006.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang.

AgentBench: Evaluating LLMs as Agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

Yichuan Ma, Linyang Li, Yongkang Chen, Peiji Li, Xiaozhe Li, Qipeng Guo, Dahua Lin, and Kai Chen. Timely Machine: Awareness of Time Makes Test-Time Scaling Agentic. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4619–4636. Association for Computational Linguistics, 2026. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.211. URL https://aclanthology.org/2026.acl-long.211/.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for General AI Assistants. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/25ae35b5b1738d80f1f03a8713e405ec-Paper-Conference.pdf.

Yuanhao Qu, Yingzhou Lu, Xinming Tu, Serena Zhang, Tianwei She, Alexander Glenn Shaw, Jou-Ho Shih, Bingqing Zhao, Minjie Shen, Haochen Yang, Jielin Yan, Rongchuan Zhang, Xinze Wu, Tingting Li, Bin Zhou, Ning Wang, Adam Ma, Le Cong, Xiaobo Hu, Yuan Jiang, Jiayun Dong, Tao Peng, Jure Leskovec, and Kexin Huang. BiomniBench: Process-level evaluation of LLM agents for real-world biomedical research. *bioRxiv*, 2026. doi: 10.64898/2026.05.12.724604. URL https://www.biorxiv.org/content/10.64898/2026.05.12.724604v2.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM Agents Automate LLM Post-Training?, 2026. URL https://arxiv.org/abs/2603.08640.

Neil K. R. Sehgal, Sharath Chandra Guntuku, and Lyle Ungar. Real-Time Deadlines Reveal Fragile Temporal Adaptation in LLM Strategic Dialogues, 2026. URL https://arxiv.org/abs/2601.13206. 2026 Conference on Empirical Methods in Natural Language Processing; arXiv:2601.13206.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, et al. Agents' Last Exam. arXiv preprint, 2026. URL https://arxiv.org/abs/2606.05405.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. Towards Benchmarking and Improving the Temporal Reasoning Capability of Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14820–14835. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.828. URL https://aclanthology.org/2023.acl-long.828/.

TeachArena authors. TeachArena: A benchmark for AI tutoring agents across situated tutoring, pedagogical judgment, and teaching workflows. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/CinderD/TeachArena.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Austin Bridges, Jesse Boyle, Koby Twist, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents. arXiv preprint, 2026. URL https://arxiv.org/abs/2601.14242.

Minghan Wang, Ye Bai, Thuy-Trang Vu, Ehsan Shareghi, and Gholamreza Haffari. Discrete Minds in a Continuous World: Do Language Models Know Time Passes? In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 18703–18729, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1016. URL https://aclanthology.org/2025.findings-emnlp.1016/.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. BudgetThinker: Empowering Budget-aware LLM Reasoning with Control Tokens, 2025. URL https://arxiv.org/abs/2508.17196.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Holden Karnofsky, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts, 2024. URL https://arxiv.org/abs/2411.15114. arXiv preprint arXiv:2411.15114.

Yue Yuan and Xiaomin Sun. Urgent tasks are concrete, important tasks are abstract: A construal level theory perspective. *Personality and Individual Differences*, 241:113191, 2025. doi: 10.1016/j.paid.2025.113191. URL https://www.sciencedirect.com/science/article/pii/S0191886925001539.

Yikai Zhang, Siyu Yuan, Caiyu Hu, Kyle Richardson, Yanghua Xiao, and Jiangjie Chen. TimeArena: Shaping Efficient Multitasking Language Agents in a Time-Aware Simulation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3894–3916. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.215. URL https://aclanthology.org/2024.acl-long.215/.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996. doi: 10.1609/aimag.v17i3.1232. URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1232.

Table 4: Distribution of AgentTime's 82 tasks by domain and source.

Domain	LHTB	ALE	Other sources	Total	Share (%)
Engineering	9	4	0	13	15.9
Physical sciences	5	3	0	8	9.8
Life sciences	2	0	6	8	9.8
Health and medicine	2	2	6	10	12.2
Business and finance	3	0	0	3	3.7
Law	1	0	12	13	15.9
Visual media	3	3	0	6	7.3
Computer science and mathematics	15	0	0	15	18.3
Education and information	0	0	6	6	7.3
Total	40	12	30	82	100.0

A Dataset and Evaluation Protocol

This appendix specifies the task inventory, execution protocol, measurement rules, and reporting requirements for Section 3. Appendix B documents reproducibility and data availability; Appendix C gives the prompt templates.

For one run, we use the notation from Section 3.3: P is the initial plan, E the full execution trace, T the scoring budget, and S the normalized final task score. All durations use the charged clock defined in Appendix A.3, with consistent units within each calculation.

A.1 Task Inventory and Screening

Section 3.1 reports the construction rationale. Table 4 gives the distribution of 82 tasks across nine professional domains by source; independently rounded shares may not sum to 100%. A task is a workflow objective, an instance fixes its inputs, environment, and evaluator, and a run executes that instance with a given agent and budget condition. Source-dataset sizes and repeated runs do not increase the number of independent tasks. Task identifiers, screening records, and instance and run counts are part of the reproducibility records listed in Appendix A.2; their current availability is stated in Appendix B.

Sources and adaptation records. The task collection draws on six sources: LHTB (Li et al., 2026b), ALE (Sun et al., 2026), BiomniBench-DA (Qu et al., 2026), PhysicianBench (Liu et al., 2026a), APEX-Agents (law) (Vidgen et al., 2026), and TeachArena (TeachArena authors, 2026). Each task record must retain the source identifier, version and citation, original assets, adaptations, artifact requirements, progress-sensitive grader, and validation records. Each task must pass screening individually; the table groups the four smaller source collections under Other sources.

Per-task checklist. Before inclusion, each task requires: (i) a source identifier and version; (ii) a prompt and accessible assets; (iii) a restorable environment; (iv) a declared artifact and explicit-submission contract; (v) a task-quality grader with score direction, baseline, and reference anchor; (vi) evidence that partial quality can be evaluated; (vii) scenario and metric eligibility; and (viii) a licensing and access record. Variants of the same task share a template identifier to avoid counting duplicates as independent tasks.

Screening combines static task analysis with observations of Qwen-code (Qwen-3.8-flash) trial runs, retaining tasks with non-binary scoring, multiple solution paths, and scope for continued improvement, such as robustness requirements. The execution criterion retains tasks that this configuration does not complete within a 30-minute screening budget. Each screening record must identify the agent, model snapshot, framework version, reasoning settings, machine allocation, prompt, runtime, completion criterion, and artifact. These runs remain separate from evaluation runs. Selection cannot depend on a favorable response to a change in the time budget.

The audit distinguishes difficult tasks from broken environments, unavailable resources, inconsistent graders, and impossible requirements. Grader checks use identical frozen artifacts; environment checks use repeated resets. Any source policy that forces continued execution must be documented

Table 5: Required records linking measurements to runs, plans, events, and final artifacts.

Record	Required fields
Task manifest	Task, instance, and template IDs; source/version; domain; inputs and environment hashes; submission and continuation contract; grader/version; score normalization; subtask boundaries; license and audit records.
Run configuration	Run and matched-group IDs; system/model versions; repetition; T1/T2/T3 condition; budget; notification schedule; tools; inference settings; hardware; Simple or Instructed prompt; continuation mode.
Timing ledger	Monotonic and UTC start; raw elapsed time; evaluator verification intervals; charged elapsed time; current total budget; notice receipt; time-budget exhaustion and termination reason.
Plans and events	Initial plan; subtask IDs and prospective subtask time estimates; timestamped plan updates; action start/end; tool requests/results; verification feedback; error provenance.
Artifact ledger	Artifact hash; recovery and receipt times; test or final submission; validity; final score and grader version.
Measurement record	S, eligible A1/A2; A3/A4 judge and unit IDs, Rating, Evidence, and calibrated scores; coverage and missingness; four run IDs and final scores used for A5.

and harmonized before stopping behavior can be interpreted. Excluded tasks retain their identifiers, reasons, and supporting evidence. The audited inventory reports coverage by source and domain, including domains left without evaluable tasks.

Submission modes and verification pauses. Following the single-shot and continuation settings of the LHTB evaluation harness (Li et al., 2026b),[1] our dataset includes both non-continuation and continuation tasks. Non-continuation tasks accept one final submission. Continuation tasks also accept <TEST_SUBMISSION>, which requests evaluator verification and feedback while time remains, allowing the agent to revise its solution and submit again. In both modes, <FINAL_SUBMISSION> binds a recoverable artifact to the controller's receipt and ends the run; later revisions cannot replace it. Matched runs use the same continuation mode, verification feedback, and evaluator version.

To prevent evaluator verification after an intermediate submission from consuming the agent's execution budget, we introduce a controller that pauses both the agent and the budget clock during verification and resumes them when verification finishes. Agent-run tests, ordinary tools, and model inference still consume the budget. Appendix A.3 formalizes the clock and time-budget enforcement rules, and Appendix C.1 gives the submission instructions.

A.2 Frozen Configuration and Required Records

The evaluation protocol fixes each agent–instruction configuration across matched runs and requires the records listed in Table 5. These include the model snapshot, framework commit, task-instruction variant, system prompt, tool schemas, context policy, hardware, concurrency, and observable service latency. Content hashes identify the assets. The task rubric, continuation mode, subtask boundaries, timing instructions, and T3 notification schedule are fixed before systems are compared.

The four evaluated configurations are listed in Section 4.1. Their reproducibility records must identify the exact model/provider snapshot, agent release and commit, reasoning and sampling settings, clock interface, context and retry policies, resource allocation, execution region and dates, and final-submission procedure. Family labels alone do not identify immutable snapshots; inaccessible vendor defaults are recorded as unknown. API throughput measurements must state the token-counting convention, measurement interval, and configurations covered. Restricted assets may be released through an authorized reconstruction procedure. Plans and estimates retain their histories: revisions cannot replace the initial plan used for A3 or alter the prospective estimates used for A2.

[1] https://github.com/zli12321/LHTB/blob/main/harbor/README.md

A.3 Clock, Time Budget, and Submission Semantics

Let t_0 be the controller's monotonic time immediately before the first model request. Let $v(t)$ be the cumulative time in controller-authorized evaluator verification intervals up to t, including the elapsed portion of an active interval. The agent is paused throughout those intervals. Raw elapsed time is $t - t_0$; charged elapsed time is

$$e(t) = t - t_0 - v(t). \tag{8}$$

Model inference, ordinary tools, agent-run tests, waiting, and retries consume the budget. Only designated evaluator verification in continuation tasks is excluded. Environment provisioning occurs before t_0, and final offline grading follows termination. Both clocks and every excluded interval are logged. The agent cannot act during excluded evaluator verification. Overlapping ordinary tools consume elapsed wall-clock time, without adding the time used by each tool.

The controller enforces $e(t) \leq T(t)$, where $T(t)$ is the current total budget. Excluded evaluator verification pauses the charged clock and delays time-budget exhaustion in wall-clock time. The corresponding wall-clock time is $D(t) = t_0 + T(t) + v(t)$. Any displayed UTC estimate of time-budget exhaustion is updated after evaluator verification or a budget change, while enforcement always uses charged elapsed time. The time interface returns current UTC time, raw and charged elapsed time, remaining charged time, total budget, and expiry status. A1 uses charged elapsed time at submission; A2 uses charged elapsed time between subtask boundaries.

The submitted artifact must be fully written and recoverable at receipt; a promised result or unfinished write is invalid. Let t_{final} be the controller's receipt time for the final submission. Receipt is on time when $e(t_{\text{final}}) \leq T$. A valid artifact may contain only a partial solution and earn zero or partial credit. When the time budget is exhausted, the controller rejects new actions and terminates the run without promoting a test submission to final. Missing, invalid, or late final submissions receive $S = 0$ and A1$= 0$ wherever A1 applies. Under the source definition, a valid on-time zero-score submission has A1$= 1$. Evaluator failures leave scores missing.

A.4 Scenario Protocol and Prompt Use

T1 assigns 120 minutes. T2 comprises separate 10- and 30-minute runs. T3 initially assigns 60 minutes and shortens the total budget to 45 minutes during execution. T1/T2 provide A2, A3, and A4; T2/T3 provide A1. All four final scores contribute to A5. Task inputs, tool access, submission mode, and scoring rules are fixed within each matched four-run group.

The task instructions supply the budget in seconds, require an initial plan with a time allocation for each subtask, and provide `python .ale/time_budget.py` to inspect the clock. Codex (GPT-5.6-sol) is evaluated with both the Simple prompt (Appendix C.1.1) and the Instructed prompt (Appendix C.1.2); Qwen-code (Qwen-3.8-flash) and DeepSeek Harness (DeepSeek-v4-pro) use the Simple prompt. Each agent–instruction configuration is held fixed across its matched four-run group. Appendix C.1 also gives the continuation instruction appended for tasks that support evaluator verification. Non-continuation tasks accept only the final submission. The release retains the exact prompt, command, and tool schema used in each run.

The interface distinguishes evaluator verification from agent-run testing, which consumes the budget. Each task contract specifies the non-continuation output format and the artifact bound to either submission marker.

The Codex prompt comparison holds the task instances, model and harness settings, budgets, tools, continuation rules, and evaluators fixed while varying the Simple versus Instructed prompt. It measures the combined effect of the additional instructions: an explicit timeout-failure statement, periodic time checks, and adjustment of the remaining plan and execution strategy when progress deviates.

Initial planning in T1/T2. Both scenarios use the same planning protocol. Before task execution, the agent records the required work, subtask IDs, subtask time estimates, initial time allocations, and any time reserved for checking or delivery. Planning consumes the run budget. The task manifest fixes the alignment and timing-boundary rules used to map each agent's planned subtasks to the execution trace; nested stages are counted only once. Cross-agent subtask calibration additionally

requires a mapping to common task–subtask units (Appendix A.6). The initial plan supplies A3, prospective estimates supply A2, and timestamped updates support A4. Estimates are collected within these runs, without a separate evaluation arm.

T3 budget change. At 30 minutes of charged elapsed time, halve the remaining budget from 30 to 15 minutes, changing the total budget from 60 to 45 minutes. Deliver the budget-update prompt in Appendix C.1, substituting the remaining charged time in seconds. The log records when the notice becomes available. Remaining time is $45\,\text{min} - e(t)$. T3 supplies A1 and the 45-minute input to A5; A4 is evaluated in T1/T2. Runs completed before the notice retain their T3 assignment and use the final 45-minute budget for A1. Their count and the actual notice timing are reported.

A.5 Measurement Details and Eligibility

Task score. The task rubric and mapping to $S \in [0, 1]$ are fixed independently of the evaluated systems. Report raw scores, reference anchors, and any clipping; the best tested agent cannot serve as the reference. Reverse lower-is-better source scores before normalization. Only the final submission determines S. Evaluator errors leave measurements missing, while observed agent failures receive zero.

A1: Punctuality. Equation 3 applies to T2/T3. The aligned subtask times cover all charged time from run start to final submission, including initial planning, with overlapping tool execution counted only once. For an on-time valid final submission, A1$= (\sum_{i=1}^{n} t'_i)/T$ when $S \neq 0$ and A1$= 1$ when $S = 0$; a late, invalid, or missing final submission receives zero. Thus A1$= 1$ can describe either a zero-score on-time result or a positive-score result submitted exactly when the time budget is exhausted. It must be read alongside S, without a preferred-direction arrow. Missing timing records make A1 unavailable.

A2: Subtask Time Estimation. For a fixed run with budget T, let m be the number of completed eligible subtasks, indexed here by $i = 1, \ldots, m$. For each, retain its prospective time estimate $\widehat{t}_i$, ID, estimate timestamp, start/end events, and actual subtask time t'_i measured on the charged clock. Compute

$$A2 = \sum_{i=1}^{m} \left| \frac{t'_i - \widehat{t}_i}{T} \right|. \tag{9}$$

Errors use the same time units and are normalized by the total budget, with no division by m or individual actual subtask times. A zero estimate contributes t'_j/T. A completed subtask with zero charged elapsed time is valid if the frozen boundary rules support it. A2 is nonnegative and has no stipulated upper bound. Subtasks with missing estimates or boundaries, and those unfinished when time runs out, are excluded with recorded reasons. Report the proportion with recorded estimates, the number included in A2, and the proportion unfinished when time runs out. A2 is unavailable when no unit is eligible. Because this metric sums errors across subtasks, subtask granularity and alignment must be documented when comparing systems; differences in plan decomposition can affect the score.

A5: Time-Use Efficiency. For each matched set of runs, let T_k denote the observed scoring budgets in increasing order, starting at $T_0 > 0$ and ending at $T_{\max}$, and let $S(T_k)$ be the final task score at budget T_k. The short-budget score $S_{T_0} = S(T_0)$ is the baseline. With b denoting the time-budget coordinate, Equation 7 defines

$$A5 = \frac{1}{T_{\max}} \int_{T_0}^{T_{\max}} \frac{S(b) - S_{T_0}}{1 - S_{T_0}}\, db. \tag{10}$$

Linearly interpolating $S(b)$ between observed budget points yields the following trapezoidal expression, where the sum covers all adjacent observed budget pairs (T_k, T_{k+1}):

$$A5 = \frac{1}{T_{\max}(1 - S_{T_0})} \sum_{k} (T_{k+1} - T_k) \left(\frac{S(T_k) + S(T_{k+1})}{2} - S_{T_0} \right). \tag{11}$$

Only budgets from T_0 onward contribute to the integral; normalization uses the maximum total budget $T_{\max}$, not the interval length $T_{\max} - T_0$. Interpolation connects final scores from separate runs

and does not represent progress within one run. Scores retain their observed order, including negative gains, with no best-so-far envelope or clipping. A5 can therefore be negative when larger-budget conditions perform worse than the baseline. For $S_{T_0} < 1$, its maximum under this interpolation rule is $[T_{\max} - T_0 - (T_1 - T_0)/2]/T_{\max}$, attained when all scores after the baseline equal one.

A5 is unavailable when $S_{T_0} = 1$ because the normalization denominator is zero; report these ceiling cases separately. Instrumentation, infrastructure, or grading failures that leave an input missing also make the matched set unavailable. No point is dropped or replaced with zero. Observed agent failures with zero final scores remain valid inputs. Retain all scores, run IDs, and missingness reasons. Compute A5 for each matched set before averaging repetitions, instances, and tasks.

Our scenarios instantiate this definition at $(T_0, T_1, T_2, T_3) = (10, 30, 45, 120)$ minutes, giving a maximum A5 of $5/6$ under linear interpolation. The 45-minute result comes from T3 and includes a budget change. A5 therefore describes this set of scenarios and cannot isolate the causal effect of extra fixed execution time.

A.6 A3/A4 Rating Eligibility and Calibration

A3 assesses time allocation and scheduling in the initial plan; A4 assesses adjustments to subtask selection, order, duration, or tool-use methods when the remaining time budget no longer matches the current plan. Both apply to T1/T2. The full prompts appear in Appendices C.2 and C.3. Three LLM judges independently return a **Rating** and supporting **Evidence**. The four ordered categories are *Clear Deficiency*, *Potential Deficiency*, *Generally Reasonable*, and *Well Supported*, coded as 0, 1, 2, and 3, respectively; *Unable to Determine* (UTD) is recorded separately.

Evidence and eligibility. A3 uses task requirements, public scoring criteria, tool instructions, the total budget, and the initial plan. A recorded but vague plan may receive a deficiency rating; an unavailable plan receives UTD. A4 additionally uses timestamped execution records, tool calls, and any recorded plan updates to assess responses to a budget-plan mismatch. A4 receives UTD when no relevant mismatch is observed, or when the available plans, timing records, or execution evidence cannot establish whether an adjustment was needed, possible, or implemented. A claimed adjustment requires evidence of changed execution. Final quality and inferred private reasoning cannot replace the required evidence, and ratings use only information available at the decision time.

Input records and evaluation units. The grading record links each invocation to its run, judge, metric, budget, and evaluation unit: a subtask allocation for A3 or the full execution trace for A4. Each judgment retains the task and plan context. For A4, PLANS supplies the initial plan, and TRACE retains any timestamped plan updates needed to reconstruct the plan active at each decision; each judge returns one Rating for the entire trace. The A3 formula requires a separate rating for each subtask allocation. Its scoring wrapper must therefore identify the target subtask for each invocation and retain the mapping to common task–subtask units. The available prompt template contains only the full plan and returns one Rating; verification of the wrapper is an outstanding reproducibility item (Appendix B). Appendix C specifies the input fields for each template.

Calibration and validation. Let $Y_{rij} \in \{0, 1, 2, 3\}$ be judge j's rating of evaluation unit i in run r, where $j \in \{1, 2, 3\}$. For A3, i indexes subtask allocations; for A4, the full execution trace is the sole unit, $i = 1$. Let $a(r)$ identify the agent configuration and $q(r, i)$ identify the matched task–subtask unit for A3 or task instance for A4. The many-facet Rasch model (Linacre et al., 1990) is

$$\log \frac{\Pr(Y_{rij} = k)}{\Pr(Y_{rij} = k - 1)} = \theta_{a(r)} - \delta_{q(r,i)} - \gamma_j - \beta_k, \tag{12}$$

where $\theta_{a(r)}$ is agent ability, $\delta_{q(r,i)}$ is matched-unit difficulty, γ_j is judge severity, and β_k is the adjacent-category threshold for $k \in \{1, 2, 3\}$. The adjustment targets severity differences and does not eliminate shared judge errors.

A3 and A4 are fitted separately within each budget using the pooled ratings from all three judges. For each rated unit, the calibrated expected category score at reference judge severity $\gamma = 0$ is divided by three to obtain a value in $[0, 1]$. A3 averages the calibrated subtask scores; when only

some subtasks can be rated, the denominator is their count rather than the full n. A4 directly uses the calibrated score for the full trace. Raw ratings and cited evidence are retained alongside the calibrated scores. UTD judgments are excluded from ordinal calibration; units without rateable evidence are not assigned zero or filled with a model prediction. A3 coverage is the proportion of subtasks rated, while A4 coverage is the proportion of runs rated. A run with no rateable units has no score for that metric. Reproduction also requires the judge versions, model identification constraints, fit diagnostics, and minimum evidence rule for a calibrated unit score (Appendix B).

Where feasible, graders are blinded to system identity and final scores. Validation requires a stratified, double-annotated pilot with reported agreement and disagreements, followed by adjudication and a frozen rating protocol. Automated graders must be checked against that sample.

A.7 Aggregation, Failures, and Reporting

For S, A1, and A2, average repetitions within each instance, instances within each task, and tasks within each applicable condition, giving tasks equal weight. Compute A5 within each matched set of runs before applying the same hierarchy. Apply the same hierarchy to the separately calibrated A3/A4 run scores. Also aggregate raw category indicators in this order to report rating distributions and UTD proportions. Raw ordinal codes are not averaged directly into scores. The reporting protocol requires eligible counts and missingness for every result. Uncertainty estimates should resample tasks or shared templates while keeping repetitions and matched conditions together.

Termination records distinguish final submission, voluntary stopping without submission, timeout, agent error, infrastructure error, and evaluator error. Observed agent failures remain outcomes. Infrastructure exclusions require evidence under rules fixed before evaluation. A repaired rerun receives a new ID linked to the failed run; regrading the same artifact does not count as another attempt. The group-to-run mapping must identify all A5 inputs and support reconstruction of every reported denominator.

Together, these records allow the numerical metrics to be recomputed and each A3/A4 rating to be checked against its evidence and calibration record. Qualitative cases follow a stated selection rule. A1–A5 remain separate because they differ in meaning, units, applicability, and output type.

B Reproducibility and Data Availability

Reported measurements. Tables 2 and 3 contain the available configuration-level results for T1, T2, and T3. Fixed-budget task scores and A3/A4 are reported for all four configurations, including Codex at 120 minutes ($S = 0.6056$, A3$= 0.7388$, A4$= 0.6311$); its T1 A2 is 0.5761. A1 is not computed under the ample T1 budget because final submission is assumed. A2 is unavailable for Codex (Instructed) in T1/T2 because replanning prevents direct alignment with the initial time estimates. These entries are shown as dashes, not zeros. A1 is evaluated in T2/T3; A2–A4 are evaluated where applicable in T1/T2.

Tables and figure. The overview-table scripts generate Tables 2 and 3 from the supplied configuration-level aggregates in `data/reported_results.tex`. Figure 3 uses `scripts/generate_figure3_headroom.py`, which reads the S column of Table 2 for all four configurations. The inputs and calculations are exported to `data/figure3_task_scores.csv`. Let $\overline{S}_b$ denote the mean task score at budget b minutes. The plotted gain is $g(b) = (\overline{S}_b - \overline{S}_{10})/(1 - \overline{S}_{10})$, with linear interpolation between separate runs at 10, 30, and 120 minutes. The figure's area annotation is

$$A5_T = \frac{1}{T}\int_{10}^{T} g(b)\,db, \qquad T \in \{30, 120\} \text{ minutes.} \tag{13}$$

Here, $A5_T$ is a descriptive summary of aggregate fixed-budget scores, distinct from the matched-group A5 in Section 3.3. All four curves include 120-minute scores. Codex's normalized 120-minute gain is $(0.6056 - 0.3907)/(1 - 0.3907) = 0.3527$, giving $A5_{30} = 0.0131$ and $A5_{120} = 0.1502$. The corresponding $A5_{120}$ values are 0.2246 for Codex (Instructed), 0.2823 for Qwen-code, and 0.2397 for DeepSeek Harness. These curves assume comparable task samples and weights across T1/T2; they exclude the T3 budget-change condition and do not show progress within a run. The protocol-level A5 instead normalizes each matched four-condition group, including T3, before aggregation. Normalizing mean scores need not equal averaging group-level normalized scores, so Figure 3 cannot recover that metric or its uncertainty without matched-run records.

Run-level records. The available materials contain aggregate results and a task-manifest schema, but no completed task manifest or underlying evaluation logs. Consequently, task membership, instance and repetition counts, matched-run denominators, eligibility and missingness counts, and uncertainty intervals cannot yet be independently reconstructed. The task manifest, screening records, frozen configurations, timing ledgers, and final artifacts specified in Appendix A are required to audit these quantities. A2 additionally requires the provenance of prospective time estimates: the agent templates request time allocations, which do not by themselves establish that separate duration estimates were recorded. Trace alignment must cover initial planning, unmatched first actions, and missing initial plans without losing charged elapsed time. The reported mean API throughput of 42.32 tokens/s also requires its measurement procedure and configuration coverage.

A3/A4 scoring records. The rubric categories and prompt templates are specified in Appendix C; the reported scores lie in $[0, 1]$. Reproducing their calibration requires the three judge identities and versions, unit-level Rating and Evidence outputs, the A3 target-subtask wrapper and alignment map, model identification constraints, fit diagnostics, and the minimum evidence rule. In particular, the A3 template alone does not establish that ratings were collected separately for each subtask as required by Equation 5. Raw category distributions, UTD proportions, annotator agreement, and validation-pilot records are not available with the aggregates. For A4, a run without an observed relevant budget-plan mismatch receives UTD under the stated rubric.

C PROMPT TEMPLATES

This appendix collects the agent instructions and the A3/A4 grading prompts. Braced fields are filled with the corresponding run records. Appendix A.4 describes when each agent instruction is used; Appendix A.6 gives the grading eligibility and calibration rules.

C.1 AGENT INSTRUCTIONS

The four templates below use input descriptions and block headings for presentation. The Simple prompt is used for Codex (GPT-5.6-sol), Qwen-code (Qwen-3.8-flash), and DeepSeek Harness (DeepSeek-v4-pro); the Instructed prompt is used for Codex (Instructed) (GPT-5.6-sol). In both budget templates, `{budget_seconds}` corresponds to the placeholder `<T>`.

C.1.1 SIMPLE PROMPT

Simple Prompt

Inputs:
BUDGET: `{budget_seconds}`
CLOCK COMMAND: `python .ale/time_budget.py`

Instructions:
TIME BUDGET: You have a total time budget of `{budget_seconds}` seconds for this task.

Before starting the task, create a plan based on your time budget, specifying how much time you will allocate to each subtask.

During the task, you can check the current time, elapsed time, and the remaining time by running: `python .ale/time_budget.py`.

Output:

> Make full use of the available time to finish the task, and return your final result within the time budget.

C.1.2 INSTRUCTED PROMPT

Instructed Prompt

Inputs:
BUDGET: `{budget_seconds}`
CLOCK COMMAND: `python .ale/time_budget.py`

Instructions:
TIME BUDGET: You have a total time budget of `{budget_seconds}` seconds to complete this task. How you allocate the available time is up to you. Use all available time strategically to produce the best result you can.

Before starting the task, create a plan based on your time budget, specifying how much time you will allocate to each subtask.

You can run `python .ale/time_budget.py` to check the current time, elapsed working time, and remaining time budget. Check the time and your remaining time budget periodically as you work. If your actual execution progress differs from your plan, adjust your remaining plan and execution accordingly.

Output:

> You must proactively end the task within the time budget. If you exceed the time budget, the task will be considered failed.

Prompt for Continuation and Submission

Inputs:
MODE: Task with evaluator verification and continuation enabled.

Instructions:
You can end your turn with a response containing `<TEST_SUBMISSION>` on a separate line to request verification of your current solution. If your solution does not pass verification and time remains, you can continue working. Verification time does not count toward your budget.

Output:

> To end the task, use `<FINAL_SUBMISSION>` instead, on a separate line in your response.

Prompt for the T3 Budget Update

Inputs:
REMAINING TIME: `{remaining_seconds}`

Instructions:
Budget update: Your total time budget is now 45 minutes, replacing the original 60-minute budget. You have `{remaining_seconds}` seconds remaining under the charged clock.

Output:

> Return your final result within the updated time budget.

C.2 A3: TIME ALLOCATION AND SCHEDULING

Prompt for Time Allocation and Scheduling

Please assess the time allocation and scheduling of subtasks in an agent's initial plan.

Inputs:
TASK: `{task_description, scoring_criteria}`
TOOL: `{tool_usage_instructions}`
BUDGET: `{total_time_budget}`
PLAN: `{initial_plan}`

Evaluation Aspects:

1. Identify the key work from the task requirements and scoring criteria. Match it semantically to the plan and assess whether the key deliverables and scoring objectives are covered.
2. Assess whether decisions to prioritize, reduce, or reserve time for particular work are grounded in the task, and whether any time-allocation choices clearly conflict with task priorities. Higher-priority subtasks should generally receive relatively more time, considering their time requirements and dependencies; do not infer allocation quality from duration alone.

Evaluation Rubric:

- **Clear Deficiency:** The plan omits required work, violates constraints, or allocates time against task priorities. It may also be too vague to identify key activities or connect them to specific time allocations.
- **Potential Deficiency:** Some key work may be missing, or parts of the plan and time allocation are too coarse. Some choices lack a clear basis and may leave insufficient time for essential work.
- **Generally Reasonable:** The plan covers key deliverables and scoring objectives, separates key activities, and assigns time to them sensibly. Some allocation choices still lack clear justification.
- **Well Supported:** The plan covers key deliverables and scoring objectives, with clear links between activities and time allocations. Key choices are supported by task requirements, scoring priorities, or known tool constraints; specify which choices the evidence supports.
- **Unable to Determine:** There is not enough evidence to make a judgment.

Output:

Rating: <select one rubric category>
Evidence: <list the relevant evidence, and briefly explain how it supports the rating; for Unable to Determine, state what information is missing>

C.3 A4: TIME-ADAPTIVE EXECUTION

Prompt for Time-adaptive Execution

Please assess whether an agent adapts its execution when the remaining time budget no longer matches its current plan.

Inputs:
TASK: `{task_description, scoring_criteria}`
TOOL: `{tool_usage_instructions}`
BUDGET: `{total_time_budget}`
PLANS: `{initial_plan}`
TRACE: `{timestamped_execution_records, tool_calls}`

Evaluation Aspects:
Identify when the remaining time budget no longer matches the current plan.

1. Based on the plans and execution trace, assess whether the agent adjusts subtask selection, order, or duration according to task priorities and the remaining budget.
2. Based on tool instructions and the execution trace, assess whether the agent adjusts tool-use methods within subtasks to fit the remaining budget.

Evaluation Rubric:

- **Clear Deficiency:** Despite evidence of a budget-plan mismatch and an opportunity to respond, the agent leaves necessary adjustments unmade or makes changes that conflict with task priorities. It may continue lower-priority subtasks, leave essential work insufficient time, or persist with unsuitable tool methods despite timing records or feedback showing the problem.
- **Potential Deficiency:** Adjustments to subtask selection, order, duration, or tool use appear delayed, incomplete, or poorly matched to the remaining budget. Available evidence suggests that essential work may receive insufficient time or that tool use may consume excessive time, but does not establish a clear deficiency.
- **Generally Reasonable:** The agent implements adjustments that broadly align subtask selection, order, and duration with task priorities and the remaining budget. Tool-use methods are also adapted where needed. Trace evidence supports these responses, though some choices remain weakly supported or have minor implementation gaps.
- **Well Supported:** Trace evidence clearly connects the budget-plan mismatch to timely and implemented adjustments in subtask arrangements or tool-use methods. Key choices are supported by actual progress, task priorities, remaining time, and relevant tool timing or feedback; specify which adjustments the evidence supports.
- **Unable to Determine:** No relevant budget-plan mismatch is observed, or the available plans, timing records, or execution evidence are insufficient to determine whether an adjustment was needed, possible, or implemented.

Output:

Rating: <select one rubric category>
Evidence: <list the relevant evidence, and briefly explain how it supports the rating; for Unable to Determine, state what information is missing>

Table 6: Coverage of time-awareness dimensions across benchmarks.

Dimension	TR / TB / ToT	Discrete Minds	CLPT	TicToc	TimeArena	Robotouille	Gaia2	RE-Bench	TSGym	PTBench	ALE	RTD	BAGEN	Timely-Eval	**AgentTime**
A1 Punctuality	–	✓^{s}	–	–	○^{s}	–	○^{s}	–	–	○	○	✓	–	✓	✓
A2 Subtask Time Estimation	–	○	○	–	–	–	–	–	–	–	–	–	○	–	✓
A3 Time Allocation and Scheduling	–	–	–	–	✓^{s}	○^{s}	○^{s}	–	○	–	–	○	–	○	✓
A4 Time-adaptive Execution	–	✓^{s}	–	○	○^{s}	–	○^{s}	–	–	–	–	○	○	○	✓
A5 Time-Use Efficiency	–	–	–	–	–	–	–	✓	○	✓	○	✓	–	✓	✓

✓: direct; ○: partial or secondary; –: not evaluated; s: simulated-clock analogue.

D Coverage of Time Awareness in Existing Benchmarks

Table 6 maps evaluation coverage to A1–A5 (Table 1). Direct coverage need not use AgentTime's exact metric; partial coverage includes qualitative or proxy evidence. A budget alone does not establish A1 coverage, nor does a recorded trajectory establish A3/A4 coverage. AgentTime entries describe protocol coverage, not experimental performance.

Time reasoning and targeted probes. TR, TB, and ToT denote TempReason, TimeBench, and Test of Time, which evaluate time information in task content (Tan et al., 2023; Chu et al., 2024; Fatemi et al., 2025). Discrete Minds reports timeouts and stepwise reasoning length as simulated time runs out; its dialogue-duration judgments provide indirect A2 evidence (Wang et al., 2025). CLPT (*Can LLMs Perceive Time?*) evaluates response- and task-level estimates, without estimating each planned subtask (Garikaparthi, 2026). TicToc evaluates information refreshing after a time gap, rather than adjustment to a remaining execution budget (Cheng et al., 2026).

Planning environments. TimeArena evaluates scheduling under simulated durations and resource constraints; bounded completion time provides partial A1 evidence, and analyses of rigid plans and feedback-dependent actions provide a qualitative A4 proxy (Zhang et al., 2024). Robotouille analyzes asynchronous prioritization; its generic failure-recovery analysis does not establish A4 coverage (Gonzalez-Pumariega et al., 2025). Gaia2 checks action timing under a continuously advancing simulated clock rather than final-submission punctuality (partial A1); scheduling and time-triggered action choices provide partial A3/A4 evidence (Froger et al., 2026). Its monetary-budget comparisons do not establish A5 coverage.

Long-horizon work. RE-Bench and PTBench (PostTrainBench) compare performance across time budgets (Wijk et al., 2024; Rank et al., 2026). TSGym (TimeSeriesGym) varies time and step budgets together (partial A5); its qualitative analysis of rushed exploration provides partial A3 evidence (Cai et al., 2025). ALE (Agents' Last Exam) relates scores to realized runtime across configurations, without a controlled budget sweep (Sun et al., 2026). PostTrainBench's time-utilization analysis and ALE's timeout analysis provide secondary A1 evidence.

Time-budget and resource studies. RTD and Timely-Eval report on-time outcomes and performance across time budgets (Sehgal et al., 2026; Ma et al., 2026). Pacing, reasoning-effort, and time-feedback analyses provide partial A3/A4 evidence, without explicit ratings of subtask allocations or budget–plan mismatches. BAGEN evaluates remaining-resource forecasts and feasibility-based stopping; token costs and simulated operational time provide A2/A4 proxies (Lin et al., 2026). A5 coverage does not imply use of AgentTime's headroom-normalized score.