Engineering and Technology Journal e-ISSN: 2456-3358

Volume 11 Issue 09 September -2026, Page No.- 11398-11412

DOI: 10.47191/etj/v11i09.05, I.F. – 8.595

Estimating Avoidable Harm for Test-Time Compute Allocation in Long-Horizon Tool-Using Language Agents: A Controlled Computational Study and External Validation Protocol

Prudvi Saisaran Ponduru[1], Pavani Priya Vyshnavi Nandanavanam[2], Sai Kesav Kumar Ponduru[3]

[1,2,3]Researcher

ABSTRACT - Long-horizon tool-using language agents must decide not only what action to take, but how much inference-time computation to spend before an action whose consequences may be difficult to reverse. Existing adaptive test-time methods allocate compute using uncertainty, planning need, budget state, or task consequence, while recent counterfactual work shows that additional computation can help in one state and harm in another. This paper studies a narrower quantity: avoidable harm, defined as action consequence multiplied by the estimated reduction in failure risk obtainable from additional computation. We formulate discrete compute allocation as a budget-constrained sequential decision problem and estimate counterfactual failure risk from randomized calibration data. In a controlled stochastic-agent study with 2,500 calibration trajectories and 6,000 paired test trajectories of horizon 20, a learned avoidable-harm allocator at the same 40-unit compute budget reduced catastrophic trajectory failures from 27.55% under uniform scaling to 19.95% (difference -7.60 percentage points; 95% paired-bootstrap CI [-8.33, -6.87]) and reduced consequence-weighted failure from 6.618 to 5.887. Relative to consequence-only routing, the method further reduced catastrophic failure by 0.62 percentage points and weighted failure by 0.274 while increasing task success by 1.10 percentage points. Across 20 independent replications, the safety and weighted-loss improvements remained significant after Holm correction. The result is not an accuracy win: task success remained lower than uniform scaling (7.85% versus 10.55%), exposing a safety-utility trade-off rather than a universal dominance claim. Distribution-shift and ablation studies further show that persistent risk state and verifier signals are not uniformly beneficial. The study therefore supports avoidable harm as a useful allocation target while explicitly limiting its claims to controlled computation; we provide a pre-registered external-validation protocol for WebArena, ToolHaystack, LongCLI-Bench, OS-Harm, and related environments.

KEYWORDS - agentic AI, test-time compute, tool-using agents, risk-sensitive decision making, adaptive inference, AI safety, long-horizon agents, counterfactual value of computation

I. INTRODUCTION

Tool-using language agents interleave reasoning with actions that modify external environments. ReAct established a canonical reasoning-and-acting pattern [1], while self-consistency and modern test-time-compute scaling showed that additional inference can improve reasoning when samples, search, or verification are used selectively [2], [3]. Recent agent-specific work has made allocation itself a research object: CATTS routes per-step compute using uncertainty [4], Learning When to Plan learns when explicit planning is useful [5], BATS allocates under token/tool budgets [6], and consequence-aware reasoning work shows that unequal error costs can justify unequal inference effort [7]. Direction-aware adaptation further cautions that a single confidence signal may not have the same operational meaning for every action or intervention [8]. The important question is therefore no longer whether extra compute can help, but which actions deserve it and why.

Reliability and safety research imposes a second constraint: additional computation is useful only if it reduces consequential failures rather than merely increasing tokens. CCPO combines reliability constraints with cost optimization [9]; conformal risk control provides a principled foundation for controlling expected loss [10]; E-valuator and Conformal Policy Control develop sequential or policy-level statistical safeguards [11], [12]; and ToolChain-CRC extends risk control toward drifting tool-use trajectories [13]. Runtime work separately addresses irreversible effects and persistent safety state through transactional execution and cross-iteration monitoring [14], [15]. At the evaluation layer, WebArena, ToolHaystack, LongCLI-Bench, OS-Harm, RiOSWorld, and VestaBench document persistent weaknesses in realistic long-horizon interaction, long-term tool robustness, programming, computer-use safety, and adversarial planning [16]-[21]. These literatures jointly motivate a compute policy that treats uncertainty, consequence, reducibility of risk, and budget as distinct variables rather than collapsing them into a single confidence score.

We therefore study action-level avoidable harm: the expected consequence-weighted reduction in failure risk obtainable by switching from a minimum compute mode to a more expensive mode. The author's prior AgentMesh-MCP work is cited only to document a previously studied governed tool-interface setting [22]; it is not used as authority for test-time scaling, causal effect estimation, uncertainty calibration, or statistical risk control, for which the primary external literature is cited. This paper asks a falsifiable question: under a fixed trajectory-level compute budget, does allocating extra inference according to estimated avoidable harm reduce severe action failures more than uniform, uncertainty-only, consequence-only, and random allocation?

The contribution is deliberately narrower than a general "safe agent framework." We do not claim the first adaptive agent, the first budget-aware agent, the first consequence-aware allocator, the first conformal agent monitor, or the first transactional safety mechanism. Instead, we isolate a testable allocation hypothesis and evaluate it with randomized compute interventions, known latent counterfactuals, common-random-number pairing, matched compute budgets, ablations, distribution shift, horizon scaling, budget sweeps, confidence intervals, replication-level hypothesis tests, and an explicit external-validation protocol. This scope is designed so that every major mechanism can fail independently and can be tested independently.

This paper makes four specific contributions:

- We define avoidable harm as consequence multiplied by counterfactual risk reduction from additional test-time computation, separating action severity from compute suitability.
- We give a budget-constrained allocation rule based on a learned potential-outcome risk model trained from randomized compute interventions, plus an oracle reference and strong matched-budget baselines.
- We run a controlled computational study with paired stochastic trajectories, ablations, compute-budget sweeps, distribution shift, horizon scaling, paired bootstrap confidence intervals, and 20 independent replication seeds.
- We report negative results explicitly: the method improves safety rather than raw task accuracy, persistence does not consistently help under shift, and monitor-only abstention can be much safer at severe utility cost. We also specify the external benchmark protocol needed before stronger claims about deployed LLM agents are justified.

II. RELATED WORK AND NOVELTY BOUNDARY

A. Test-time computation, reasoning search, and verification

The immediate foundation is inference-time reasoning. Chain-of-thought prompting made intermediate reasoning steps a practical inference primitive [23]. Repeated sampling can scale coverage over large inference budgets [24], while Tree of Thoughts and Graph of Thoughts replace a single chain with explicit search or graph-structured reasoning [25], [26]. Self-Refine and Reflexion show that test-time feedback and reflection can improve outputs without changing base-model weights [27], [28]. RAP and LATS cast reasoning or acting as explicit planning/search problems using world-model-like simulation or Monte Carlo tree search [29], [30]. Verifier-based work further shows that extra samples become substantially more useful when a verifier can rank candidate solutions or process steps [31], [32]. Together with self-consistency and compute-optimal scaling [2], [3], these studies establish a broad design space of repeated sampling, search, planning, refinement, reflection, and verification. Our method does not introduce another reasoning primitive; it decides where limited access to such primitives should be spent.

B. Tool augmentation and environment-grounded agents

Adaptive compute matters here because tool-using agents externalize model decisions into environments. Toolformer demonstrated that language models can learn to invoke external tools [33], while MRKL proposed modular routing between language models and expert modules [34]. ToolLLM and API-Bank expanded the scale and evaluation of API use [35], [36]; RestGPT and Gorilla explored REST/API interaction and large API collections [37], [38]; ToolkenGPT represented tools through learned embeddings [39]; and Chameleon and ToolQA studied compositional reasoning and tool-augmented question answering [40], [41]. These works make clear that tool use is not merely text generation: the model selects operations whose downstream state changes may have different costs and reversibility. Our single self-citation [22] is limited to governed agent-tool interfaces; the technical foundations for tool use itself are the external works above.

C. Long-horizon and real-world agent evaluation

Evaluation has increasingly moved from static question answering to interactive, stateful tasks. WebShop studies language-guided shopping interactions [42], AgentBench spans multiple interactive environments [43], and OSWorld tests agents in real computer environments [44]. WorkArena targets enterprise knowledge-work workflows [45], VisualWebArena adds visually grounded web tasks [46], and AgentBoard emphasizes fine-grained multi-turn analysis [47]. Mind2Web provides real-world web interaction data [48], while SWE-bench and SWE-agent expose long-context, execution-grounded software-engineering problems [49], [50]. ScienceWorld provides another sequential environment in which actions alter a simulated world [51]. These benchmark families complement WebArena, ToolHaystack, LongCLI-Bench, OS-Harm, RiOSWorld, and VestaBench [16]-[21]. Their common lesson is methodological: a claim about long-horizon agents should survive stateful interaction, execution feedback, and heterogeneous failure modes rather than a single static benchmark.

The present controlled study is intentionally not presented as a replacement for these benchmarks. Its purpose is to identify whether the proposed allocation principle has a causal mechanism under known counterfactual structure. Section 8 therefore specifies how the hypothesis should be transferred to public interactive environments with frozen model versions, disjoint calibration/test trajectories, matched resource budgets, executable outcomes, and safety-specific endpoints.

D. Agent security and runtime assurance

Once agents can issue tool calls, the attack surface expands beyond ordinary hallucination. ToolEmu uses an LLM-emulated sandbox to identify tool-use risks [52]. AgentDojo evaluates prompt-injection attacks and defenses in an agentic environment [53]. InjecAgent benchmarks indirect prompt injection in tool-integrated agents [54], ToolSword analyzes safety across stages of tool learning [55], and Greshake et al. demonstrated that indirect prompt injection can compromise LLM-integrated applications through untrusted external content [56]. These concerns are directly relevant to compute allocation because extra reasoning is not automatically a safety intervention: it can also amplify a compromised plan. Runtime safeguards such as Cordon, persistent safety state, OS-Harm, RiOSWorld, and VestaBench [14], [15], [19]-[21] therefore motivate treating severe environmental effects as a separate outcome, not as a proxy for task accuracy.

E. Uncertainty, selective prediction, and conformal risk control

Raw model confidence is not sufficient evidence of calibrated risk. Modern neural networks can be systematically miscalibrated [57], and language models exhibit structured but imperfect self-knowledge [58]. Semantic-uncertainty methods aggregate meaning-equivalent generations [59], semantic entropy can detect hallucinations in large language models [60], and SelfCheckGPT uses self-consistency among samples to detect unsupported generations [61]. Selective classification formalizes the option to abstain on uncertain inputs [62], and SelectiveNet jointly learns prediction and selection [63]. Conformal prediction offers distribution-free uncertainty tools under explicit assumptions [64]; adaptive conformal inference, beyond-exchangeability methods, learn-then-test risk calibration, risk-controlling prediction sets, covariate-shift conformal prediction, and the NLP conformal survey extend the discussion to nonstationarity, risk control, and language applications [65]-[70]. The distinction is central to this paper: our simulated estimator is deliberately not claimed to be conformally valid, and real-agent deployment would require calibration assumptions to be stated and tested rather than implied.

F. Constrained and risk-sensitive sequential decision making

The allocation problem is also related to safe and constrained sequential decision making. Constrained Policy Optimization provides a policy-learning approach for satisfying constraints [71], while the safe reinforcement-learning literature surveys risk, constraints, and intervention mechanisms [72]. Shielding can block unsafe actions using an external safety mechanism [73]; conditional-value-at-risk objectives provide a risk-sensitive optimization view [74]; and constrained Markov decision processes formalize sequential reward optimization under cost constraints [75]. Real-world reinforcement-learning analyses emphasize partial observability, delayed effects, nonstationarity, and safety constraints that complicate deployment [76]. Our controller differs in scope: it does not train the base agent policy. It allocates inference resources around a frozen decision process, but the CMDP and risk-sensitive literature provides the correct conceptual vocabulary for separating utility, cost, and safety.

G. Closest-work comparison

TABLE I. CLOSEST-WORK NOVELTY COMPARISON

Existing work	Core contribution	Boundary / limitation	Difference studied here
CATTS [4]	Uncertainty-driven per-step compute	Does not target consequence-weighted preventable harm	Counterfactual harm reduction as the routing score
BATS [6]	Budget-aware tool-use scaling	Budget state is primary allocation signal	Same hard budget; rank actions by estimated preventable harm
Wen et al. [7]	Consequence-aware compute routing	Task-level consequence; marginal utility is not the sole target	Action-level consequence × estimated risk reduction
DIAL [8]	Learns direction of compute utility	Optimizes success-cost utility, not consequence-weighted harm	Adds explicit action consequence and severe-failure endpoints
CCPO [9]	Reliability-constrained cost optimization	Different problem setting and sequence abstraction	Discrete pre-action compute intervention under long horizons
ToolChain-CRC [13]	Trajectory risk control and anytime alarms	Risk detection/intervention, not compute allocation	Risk reduction from extra compute is the decision target

Existing work	Core contribution	Boundary / limitation	Difference studied here
Cordon [14]	Transactional protection for irreversible actions	Execution containment after intent formation	Compute allocation before action commitment
Safety Does Not Compose [15]	Persistent cross-iteration safety state	Persistence itself is the contribution	Persistence is ablated as one routing feature

III. PROBLEM FORMULATION

Consider an agent interacting for horizon H. At step t it has history h_t, proposes tool action a_t, and chooses an inference mode c_t from a finite set C before executing the action. The decision is constrained by a trajectory-level compute budget B, which places the problem near constrained and risk-sensitive sequential decision making [71], [74], [75], but our intervention variable is inference effort rather than a retrained environment-action policy. Let $s_t>=0$ denote action consequence and $Y_t(c)$ in {0,1} the potential failure outcome if compute mode c were used. The conditional risk is $r_t(c)=P(Y_t(c)=1|x_t)$, where x_t contains the observable pre-action features.

$AH_t(c; c0) = s_t \cdot [\, r_t(c0) - r_t(c) \,]_+$.

Here c0 is the minimum-compute mode and $[x]_+=\max(x,0)$. If extra computation is predicted to increase risk, the avoidable-harm gain is zero rather than negative; a separate abstention or state-repair policy may be appropriate. For a trajectory budget B and compute cost $\kappa(c)$, the discrete allocation problem is

$\min_\pi \; E_\pi [\, \Sigma_{(t=1)^H} \; s_t \, Y_t \,]$ subject to $\Sigma_{(t=1)^H} \kappa(c_t) \le B$.

The formulation does not assume that task success and safety are identical objectives. A policy can reduce weighted harm while decreasing end-to-end completion, and the experiments report both. This separation prevents an unsafe success gain or an over-conservative abstention policy from being hidden inside a single scalar score.

A. Oracle ranking observation

For the special case of one optional upgrade c0→c1 with equal incremental cost Δ for every action and known conditional risks, selecting the k actions with the largest $AH_t(c1;c0)$ minimizes expected consequence-weighted failure among all k-upgrade allocations. The result follows immediately because expected weighted loss under the base policy differs from an upgraded subset only by the sum of its avoidable-harm terms. Multiple discrete tiers reduce to a knapsack-style allocation over marginal upgrades; the learned controller used here is a budget-feasible ranking heuristic rather than a claim of globally optimal discrete control.

IV. AVOIDABLE-HARM COMPUTE CONTROLLER

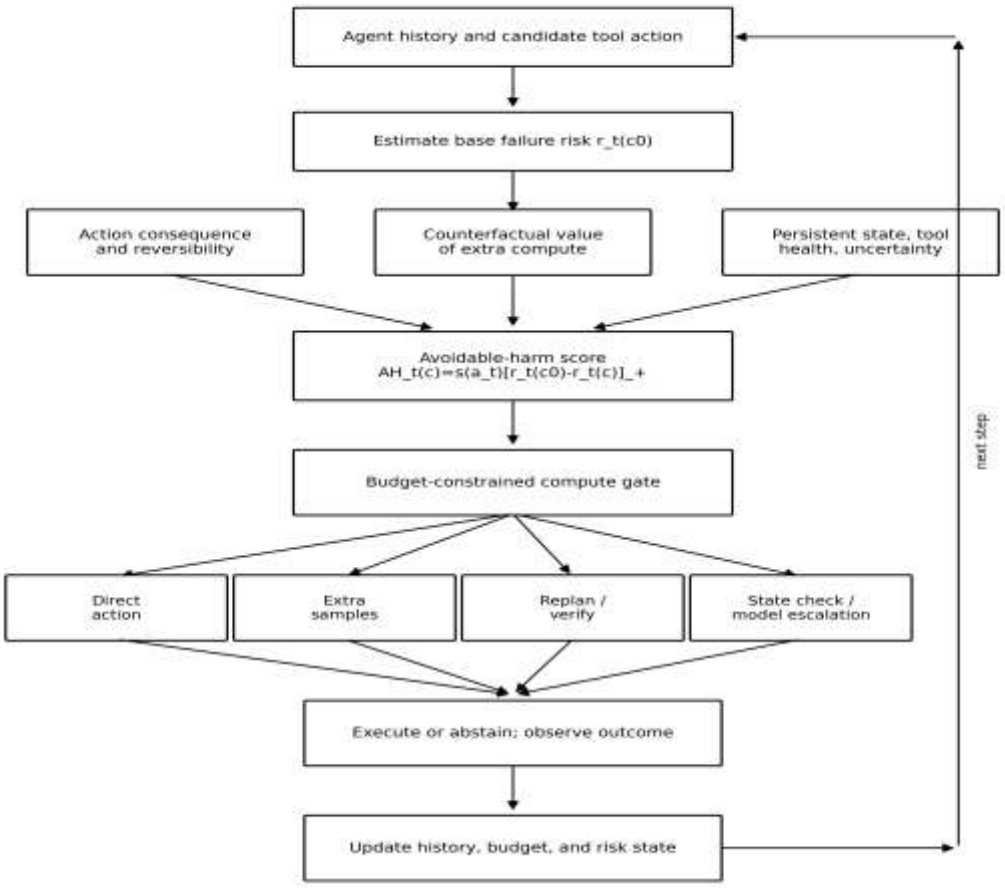

Fig. 1. Monochrome data flow of the proposed allocation rule. The controller estimates baseline failure risk, consequence/reversibility, and counterfactual benefit from additional computation, then selects a budget-feasible inference mode before execution.

A. Counterfactual failure-risk estimator

A deployment cannot observe $Y_t(c0)$ and $Y_t(c)$ simultaneously for the same action, so the target is naturally expressed in potential-outcome language [77], [78]. In the controlled calibration study, compute tiers are randomized before the action outcome is sampled. Under consistency and the generator's no-interference construction, random assignment identifies tier-specific conditional risks without observational treatment-selection bias. This is stronger than learning from a historical allocator that already chose compute based on perceived difficulty. Propensity-score methods become relevant only if future real-agent calibration relies on nonrandom logged policies rather than randomized interventions [79]. We fit a predictive risk model to the randomized intervention data and use it to estimate $r_t(c)$ for each candidate compute tier.

For each test action, the model predicts failure risk under c=1 and c=8. The learned routing score is $s_t \cdot \max(0, \hat{r}_t(1) - \hat{r}_t(8))$. Fixed calibration quantiles map approximately the top 4% of scores to tier 8, the next 14% to tier 4, the next 30% to tier 2, and the remainder to tier 1. A per-trajectory adjustment then downgrades or upgrades marginal allocations until the hard compute budget is met. Baselines use the same tier proportions and identical hard budget whenever applicable, so improvements cannot be attributed to extra compute.

B. What the method does not claim

The risk model in this study is not conformally calibrated and the paper does not claim finite-sample risk guarantees. Conformal and e-process methods are relevant future statistical layers [9-13], but ordinary i.i.d. guarantees cannot simply be transferred to adaptively changing trajectories. Likewise, the controller does not replace transactional execution [14] or persistent loop safeguards [15].

V. CONTROLLED COMPUTATIONAL STUDY

A. Generator and experimental units

The purpose of the controlled study is a proof of mechanism under known latent counterfactual structure, not a substitute for public LLM-agent benchmarks. Each trajectory contains H sequential actions with latent difficulty, tool-fault and stale-state indicators, consequence class $s \in \{1,3,10\}$, action irreversibility, persistent burden, and a latent compute-suitability variable. Consequence is generated approximately independently of difficulty so that a policy cannot win merely by treating "hard" and "important" as synonyms. Tool faults and stale state reduce compute suitability, allowing extra reasoning to be neutral or harmful in some states, consistent with the qualitative failure mode identified by DIAL [8].

For the main IID generator, the minimum-compute failure probability is

$$r_t(1) = \sigma(-3.55 + 3.15\, d_t + 1.05\, f_t + 0.80\, q_t + 0.78\, b_t),$$

where d_t is difficulty, f_t is a tool-fault indicator, q_t is stale state, b_t is persistent burden, and σ is the logistic function. Compute suitability u_t is clipped to [-0.45,1.05] and decreases sharply under tool faults or stale state. Failure probability at compute tier c is

$$r_t(c) = \sigma(\mathrm{logit}(r_t(1)) - 0.92\, u_t \log_2 c).$$

A shared uniform random variate per action is used across policies, enabling paired comparisons: a policy fails at a step if the shared variate is below its tier-specific failure probability. This common-random-number design reduces Monte Carlo noise and makes the paired bootstrap meaningful.

B. Sample sizes and compute budgets

- Calibration: 2,500 trajectories × 20 actions = 50,000 randomized action-level compute interventions.
- Main paired test: 6,000 independent trajectories, H=20, hard per-trajectory compute budget B=40, compute tiers {1,2,4,8}.
- Replication study: 20 independent seeds with 1,200 trajectories per seed and exactly matched B=40 for the adaptive baselines.
- Robustness: a shifted generator with more tool/stale-state faults, horizons $H \in \{10,20,40,80\}$, and compute-budget sweeps $B \in \{30,35,40,45,50,60\}$.
- Primary outcomes: end-to-end task success, catastrophic trajectory rate (any failed severity-10 action), consequence-weighted failure $\Sigma s_t Y_t$, any-failure rate, compute use, and abstentions.

C. Baselines and ablations

TABLE II. BASELINES AND ABLATIONS IN THE CONTROLLED STUDY

Policy	Allocation rule	Purpose
Minimal	Tier 1 at all actions	No extra inference
Uniform-2	Tier 2 at all actions	Matched 40-unit reference
Random matched	Random routing with matched budget	Controls for heterogeneous allocation alone
Uncertainty	Route by observed uncertainty	CATTS-like signal class [4]
Consequence-only	Route by consequence class	Tests whether severity alone explains gains [7]
Risk × consequence	Route by estimated base risk × severity	Strong risk-sensitive baseline
Learned avoidable harm	Severity × predicted counterfactual risk reduction	Proposed allocation target
Oracle avoidable harm	Uses true simulator counterfactual risk	Upper diagnostic bound

Policy	Allocation rule	Purpose
Monitor-only	Minimal compute; abstain on high consequential risk	Safety via refusal rather than extra compute

Ablations remove consequence weighting, persistent burden, or verifier risk from the learned score. The "no persistence" and "no verifier" variants are important because neighboring literature might otherwise encourage treating these components as automatically useful [11,15].

D. Statistical analysis

For the 6,000 paired main trajectories, we report 95% paired-bootstrap confidence intervals from 3,000 resamples; bootstrap resampling follows the classical nonparametric principle of approximating sampling uncertainty from the empirical distribution [80]. Across 20 independent replication seeds, we use one-sided Wilcoxon signed-rank tests for paired method differences [81] and control the family of primary comparisons using Holm's sequentially rejective procedure [82]. McNemar's test is reserved in the external-validation protocol for paired binary success/failure outcomes when the same task instances are evaluated by two policies [83]. Most importantly, all policies within a synthetic trajectory share the same latent action variables and the same per-action uniform random variates. This common-random-number design reduces irrelevant Monte Carlo variance and makes the paired effect interpretable as a policy contrast rather than a difference between unrelated random worlds [84], [85]. The statistical unit for the main bootstrap is the trajectory; the replication analysis uses independently seeded trajectory sets, avoiding pseudo-replication at the action level.

VI. RESULTS

A. Main matched-budget comparison

TABLE III. MAIN MATCHED-BUDGET COMPARISON

Policy	Task success	Catastrophic	Weighted failure	Compute	Abstentions
Minimal	3.45%	37.82%	9.639	20.00	0.00
Uniform-2	10.55%	27.55%	6.617	40.00	0.00
Random matched	7.22%	30.47%	7.503	40.00	0.00
Uncertainty	9.22%	28.73%	7.095	40.00	0.00
Consequence-only	6.75%	20.57%	6.160	40.00	0.00
Risk x consequence	6.80%	20.88%	6.394	40.00	0.00
Learned avoidable harm	7.85%	19.95%	5.887	40.00	0.00
Oracle avoidable harm	8.13%	18.97%	5.726	40.00	0.00
Monitor-only	0.22%	3.53%	4.087	20.00	3.57

At exactly the same 40-unit trajectory budget, learned avoidable-harm routing reduced catastrophic failure from 27.55% under uniform scaling to 19.95%. The paired difference was -7.60 percentage points (95% CI [-8.33, -6.87]). Consequence-weighted failure fell from 6.617 to 5.887, a difference of -0.731 (95% CI [-0.819, -0.642]).

The safety gain is not a free accuracy gain. End-to-end task success was 7.85% versus 10.55% for uniform scaling, a -2.70 percentage-point difference. This is the central negative result: the proposed objective changes which errors receive compute and improves severe-failure outcomes at the expense of some aggregate completion. The appropriate interpretation is a different point on the safety-utility frontier, not dominance on all objectives.

Against consequence-only routing, the learned method achieved 19.95% versus 20.57% catastrophic trajectories, while improving task success from 6.75% to 7.85% and lowering weighted failure from 6.160 to 5.887. This smaller but statistically stable increment is the more important novelty test because consequence-only allocation is a strong baseline after Wen et al. [7].

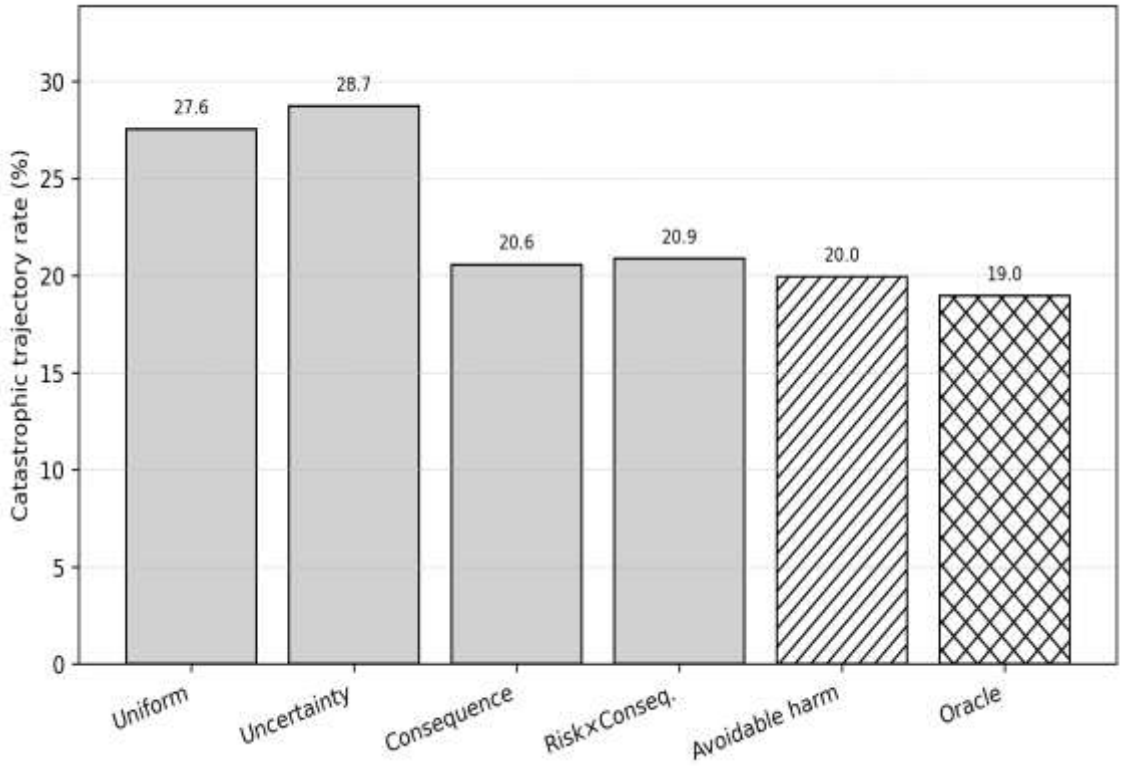

Fig. 2. Catastrophic trajectory rate in the main 6,000-trajectory matched-budget study. All adaptive policies shown use the same hard 40-unit budget; learned avoidable harm reduces severe failures beyond consequence-only and risk-times-consequence routing.

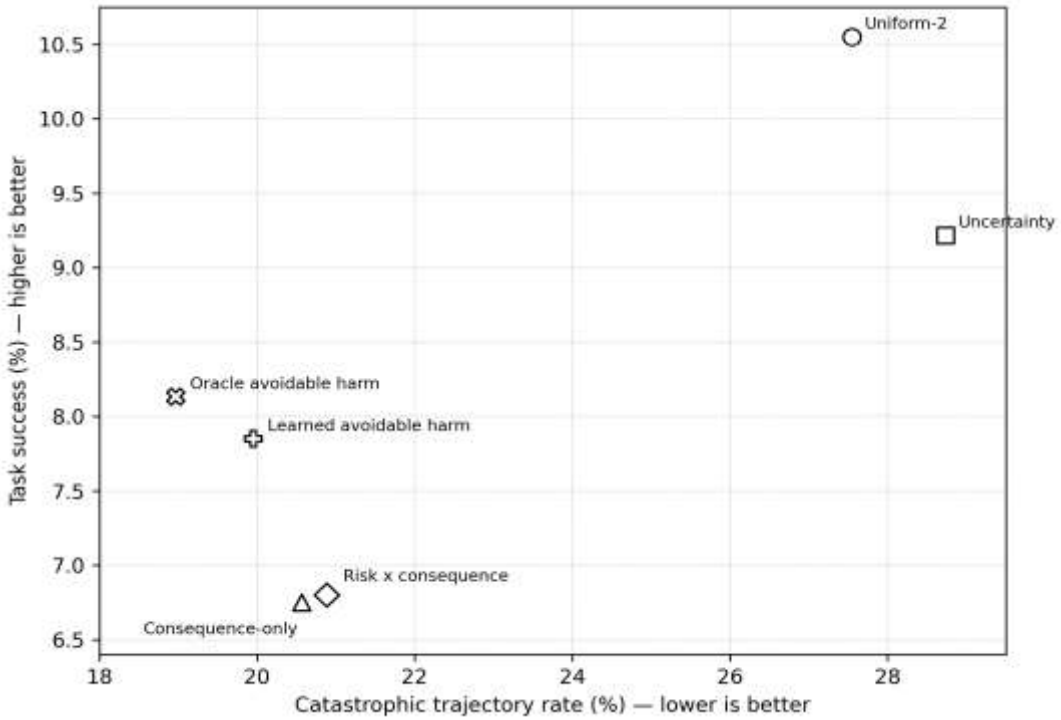

Fig. 3. Safety-success trade-off at the same compute budget. Avoidable-harm routing occupies a lower-catastrophe but lower-task-success region than uniform and uncertainty-only scaling; the oracle indicates remaining headroom in counterfactual benefit estimation.

B. Replication-level inference

TABLE IV. REPLICATION-LEVEL RESULTS

Policy	Mean success	Mean catastrophic	Mean weighted failure	Mean compute
Uniform-2	10.36%	28.21%	6.704	40.00
Uncertainty	8.91%	29.83%	7.245	40.00
Consequence-only	6.69%	20.98%	6.216	40.00
Risk x consequence	6.55%	21.57%	6.477	40.00
Learned avoidable harm	7.70%	20.30%	5.957	40.00
Oracle avoidable harm	8.10%	19.35%	5.773	40.00

Across 20 independent replication seeds, learned avoidable harm averaged 20.30% catastrophic trajectories, compared with 28.21% for Uniform-2, 29.83% for uncertainty routing, 20.98% for consequence-only, and 21.58% for risk × consequence. Holm-adjusted one-sided Wilcoxon tests were below 0.001 for catastrophic rate and consequence-weighted failure against all four baselines. Task success was significantly higher than consequence-only and risk × consequence but significantly not higher than Uniform-2 or uncertainty; the one-sided tests against the latter two were deliberately non-significant after correction.

C. Ablation study

TABLE V. ABLATION STUDY

Variant	Task success	Catastrophic	Weighted failure	Compute
Learned avoidable harm	7.85%	19.95%	5.887	40.00
Ablation: no consequence	10.65%	26.68%	6.476	40.00
Ablation: no persistence	7.77%	20.07%	5.897	40.00
Ablation: no verifier	7.23%	19.57%	5.936	40.00

Removing consequence weighting largely restored uniform-like task success (10.65%) but also restored a high catastrophic rate (26.68%). This is the clearest ablation: the method's safety behavior is driven by consequence-aware prioritization rather than generic compute adaptivity. By contrast, removing persistent burden barely changed the IID result (20.07% versus 19.95% catastrophic), and removing the verifier slightly lowered catastrophic rate (19.57%) but worsened task success and weighted failure. These mixed results argue against presenting persistence or an auxiliary verifier as necessary components.

D. Distribution shift and horizon stress

Under the shifted generator, Uniform-2 reached 35.16% catastrophic trajectories, while learned avoidable harm reached 29.08% and the oracle reached 25.72%. The no-persistence ablation reached 28.72%—slightly better than the full controller—showing that a persistent feature calibrated in one regime can be neutral or counterproductive under changed dynamics. This matters because persistence can be structurally necessary for some safety threats [15] without being automatically useful for compute allocation.

At longer horizons, absolute task success collapses for all policies because any one of many action errors can fail the trajectory. Nevertheless, severe-failure separation remains visible: at H=80, catastrophic rates were 72.88% for Uniform-2, 57.44% for learned avoidable harm, and 55.00% for the oracle. These numbers should not be read as a successful long-horizon agent—near-zero task success makes the opposite point. They show only that consequence-sensitive allocation still changes the composition of failures when the underlying agent is weak.

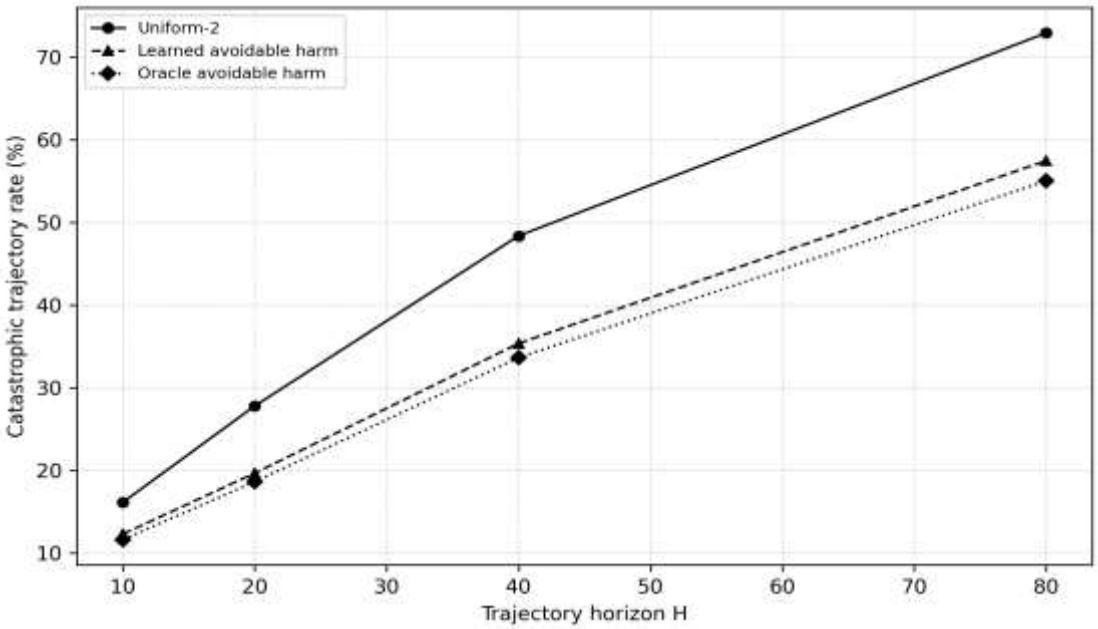

Fig. 4. Catastrophic trajectory rate versus horizon under compute budgets scaled to two units per action. Safety gains persist as horizon increases, but end-to-end task success becomes negligible at extreme horizons.

E. Compute-budget sweep

The budget sweep evaluates whether the conclusion is an artifact of B=40. Across budgets from 30 to 60, learned avoidable-harm routing maintained lower catastrophic failure than uniform scaling and generally improved on consequence-only routing. The oracle gap remained non-zero, indicating that better estimation of where computation is actually useful could matter more than adding further architectural modules.

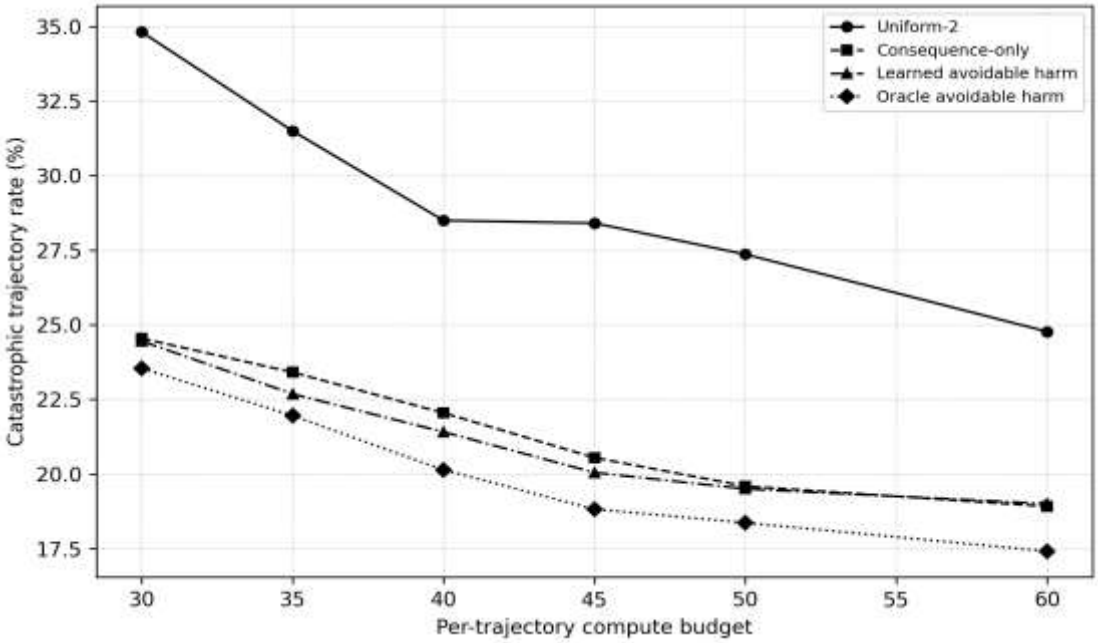

Fig. 5. Catastrophic trajectory rate over the compute-budget sweep. The learned allocator consistently prioritizes severe-risk reduction; the oracle curve quantifies remaining counterfactual-estimation headroom.

F. Monitor-only negative control

Monitor-only abstention reduced catastrophic trajectories to 3.53% but task success to only 0.22% while averaging 3.57 abstentions per trajectory. This baseline is intentionally uncomfortable: if an application values safety infinitely more than task completion, abstention can outperform reasoning allocation. A serious deployment paper must therefore specify utility, escalation, or human-approval costs rather than treating lower unsafe-action rate alone as success.

VII. DISCUSSION

A. What the experiment supports

The controlled study supports a narrow conclusion: when action consequence and the reducibility of error by extra computation are distinct latent variables, estimating their product yields a better severe-failure allocation than using uncertainty, consequence, or base risk alone. The result is strongest against consequence-only routing because that baseline already captures the central insight of recent consequence-aware compute work [7]. The oracle gap indicates that the remaining research problem is counterfactual compute-suitability estimation, not simply adding more risk features.

B. What the experiment does not support

The study does not establish that the method improves real LLM agents, that its risk estimates are calibrated under adaptive distribution shift, or that extra reasoning reliably repairs semantic tool errors. It does not establish a universal Pareto improvement because uniform scaling achieved higher raw task success. It also does not show that persistent risk state or an auxiliary verifier is universally helpful. These boundaries should be preserved in any submission and abstract revision.

C. Relation to runtime assurance

Runtime assurance and compute allocation are complementary. E-valuator and ToolChain-CRC can decide whether a trajectory should be rejected or escalated under statistical criteria [11,13], while Cordon can stage irreversible effects [14]. Avoidable-harm allocation asks a different question before commitment: whether spending additional inference is expected to avert enough consequence-weighted risk to justify its cost. A mature system may combine all three layers, but such integration is outside the empirical claim of this paper.

VIII. EXTERNAL VALIDATION PROTOCOL FOR REAL AGENTS

A top-tier archival claim requires evaluation on real interactive agents. The following protocol is specified in advance to reduce researcher degrees of freedom if this controlled result is extended.

TABLE VI. EXTERNAL VALIDATION BENCHMARKS AND MEASURES

Benchmark	Why it matters	Primary measures
WebArena / maintained subset [16]	General long-horizon web interaction	Task success, action count, tokens, cost; action consequence labels derived from side-effect class
ToolHaystack [17]	Long-term tool robustness under contextual noise	Tool failure, recovery, trajectory degradation, compute allocation points
LongCLI-Bench [18]	Very long command-line engineering tasks	Pass/progress, irreversible command errors, compute and recovery
OS-Harm [19]	Computer-use safety under misuse and prompt injection	Unsafe action completion, benign completion, abstention, severity
RiOSWorld [20]	User- and environment-originated computer risks	Risk intention/completion, task success, cross-environment transfer
VestaBench [21]	Safe long-horizon planning under constraints	Constraint violation, safe plan completion, adversarial robustness

The protocol should use at least two open-weight model families plus one independent API model if budget permits. Randomized calibration trajectories must be disjoint from the locked test set. A compact but heterogeneous main suite should cover (i) realistic web/knowledge-work interaction through WebArena or WorkArena [16], [45], (ii) long-duration tool or programming behavior through ToolHaystack, LongCLI-Bench, SWE-bench, or SWE-agent [17], [18], [49], [50], and (iii) consequential computer-use or adversarial safety through OS-Harm, RiOSWorld, VestaBench, AgentDojo, or InjecAgent [19]-[21], [53], [54]. OSWorld and VisualWebArena provide useful out-of-domain computer-use confirmation [44], [46]. Each policy must receive the same hard resource budget; outcomes should be executable whenever possible; full trajectories, tool calls, token counts, latency, and compute decisions should be logged; and safety labels should be audited independently of task completion. This design turns the external phase into a direct test of the same null hypothesis evaluated by the controlled study rather than a collection of unrelated benchmark scores.

IX. LIMITATIONS AND THREATS TO VALIDITY

- Synthetic external validity. The generator is intentionally structured and cannot reproduce semantic hallucination, prompt sensitivity, model-verifier correlation, or real tool-state dynamics. The reported numbers must not be presented as benchmark performance of an LLM.
- Counterfactual identification. Randomized compute calibration supports effect estimation in the controlled study, but real environments may make repeated rollouts expensive, non-reversible, or non-stationary.

- Consequence labels. The discrete severity levels {1,3,10} are an operational abstraction. Real consequence depends on application context, permissions, users, and reversibility.
- Adaptive dependence. The method uses a predictive risk model, not a formal conformal or anytime-valid guarantee. Sequential calibration under policy-induced shift remains open [9]-[13]. Standard conformal guarantees rely on assumptions that can fail under changing or adaptively selected trajectories; adaptive, covariate-shift, and beyond-exchangeability methods show possible remedies but also make the required assumptions explicit [64]-[70]. Any future guarantee should therefore define the exchangeability or online-validity unit and report empirical violations under model and environment shift.
- Objective trade-offs. Lower catastrophic rate can coexist with lower task success. Different applications may rationally choose a different point on the frontier or prefer human approval/abstention.
- Benchmark contamination and model drift. Future real-agent experiments must freeze model versions, prompts, environments, and judge versions, and should use executable outcomes where available. Work on real-world RL highlights nonstationarity and deployment mismatch as first-class validity threats [76], while modern agent benchmarks span substantially different observation and action spaces [42]-[51]. Cross-benchmark agreement is therefore more informative than a single aggregate score.

X. ETHICAL AND SAFETY CONSIDERATIONS

The method is intended to reduce risk from autonomous tool actions, but its evaluation can involve unsafe commands, data-exfiltration scenarios, prompt injection, or other harmful behaviors represented in modern agent-safety benchmarks [19]-[21], [52]-[56]. Such experiments should run only in isolated sandboxes or benchmark environments with no access to real credentials, external messaging, destructive file systems, or production infrastructure. Released trajectories should be reviewed for dual-use content. The method must also be evaluated for a specific failure mode: extra compute can make a malicious or compromised plan more capable. Safety is therefore measured independently of task success, and abstention/transactional controls remain valid alternatives when additional reasoning cannot reduce risk.

XI. REPRODUCIBILITY STATEMENT

The controlled study is fully specified by the generator equations, compute tiers, sample sizes, seeds, and statistical procedures reported here. The main seed is 20260830; calibration, test, replication, robustness, budget-sweep, and plotting artifacts are versioned separately. The literature in this manuscript follows claim-to-reference discipline: references are included only when they substantiate a technical premise, adjacent method, evaluation environment, safety failure mode, causal-identification choice, uncertainty/calibration assumption, or statistical procedure. The bibliography is not padded to reach a target count. Reproducibility for the external phase requires frozen model identifiers, complete prompts, environment versions, calibration/test splits, full trajectories, token/tool-call accounting, random seeds, and scripts that regenerate all primary tables and confidence intervals.

XII. CONCLUSION

Adaptive test-time compute for agents is no longer novel by itself, and consequence-aware routing is no longer an open novelty claim. The unresolved question is whether an agent can identify the portion of consequential risk that extra computation can actually prevent. This paper formalizes that quantity as avoidable harm and tests it under controlled sequential counterfactuals. At matched compute, learned avoidable-harm routing substantially reduced catastrophic and consequence-weighted failures relative to uniform and uncertainty-based scaling, and it produced smaller but repeatable improvements over consequence-only and risk-times-consequence routing. The method did not improve raw task success over uniform scaling, and several common safety features did not help consistently under shift. These negative results are part of the contribution: they show that risk-sensitive compute allocation is a trade-off problem, not a universal "more reasoning is safer" rule. The next evidentiary step is locked, cost-matched validation on real long-horizon tool and computer-use benchmarks.

APPENDIX A. CONTROLLED GENERATOR AND ALLOCATION DETAILS

A.1 Latent variables. Difficulty d_t is Beta(2.2,3.6). Tool fault f_t occurs with probability 0.08 and stale state q_t with probability 0.10 in IID experiments. Consequence s_t is 10 with probability 0.15, 3 with probability 0.35, and 1 otherwise; it does not enter the base failure-risk equation. Irreversibility is conditionally more likely for higher consequence. Persistent burden follows a clipped autoregressive process with coefficient 0.72 and innovations from difficulty, tool faults, stale state, and Gaussian noise.

A.2 Compute suitability. In the IID generator, $u_t=\mathrm{clip}(0.78-0.78\,f_t-0.72\,q_t+0.18(0.5-d_t)+\varepsilon_t,-0.45,1.05)$, $\varepsilon_t\sim N(0,0.16^2)$. Negative u_t means that extra computation increases failure probability. The distribution-shift generator raises tool-fault/stale-state probabilities and weakens the predictive value of persistent burden while reducing average compute suitability.

A.3 Observed features. Uncertainty is a noisy function of difficulty, tool faults, and stale state. Verifier risk is base failure probability plus Gaussian noise. State consistency is a

noisy proxy for tool/stale-state health. The controller never observes latent compute suitability directly.

A.4 Routing. Score thresholds are fixed from calibration quantiles. Each policy first maps its scalar score to tiers 1/2/4/8 using the same target proportions. A deterministic per-episode budget step downgrades low score-per-cost upgrades if the proposed allocation exceeds B and spends remaining budget on the highest score-per-cost feasible upgrades until no further upgrade fits. Thus key main-study baselines use exactly the same 40-unit compute budget.

APPENDIX B. PRIMARY PAIRED EFFECTS

TABLE VII. PRIMARY PAIRED EFFECTS

Baseline	Metric	Learned − baseline	95% paired-bootstrap CI
Uniform-2	Catastrophic rate	-7.60 pp	[-8.33, -6.87] pp
Uniform-2	Weighted failure	-0.731	[-0.819, -0.642]
Uniform-2	Task success	-2.70 pp	[-3.30, -2.08] pp
Consequence-only	Catastrophic rate	-0.62 pp	[-1.10, -0.17] pp
Consequence-only	Weighted failure	-0.274	[-0.335, -0.214]
Consequence-only	Task success	1.10 pp	[0.73, 1.50] pp
Risk x consequence	Catastrophic rate	-0.93 pp	[-1.38, -0.50] pp
Risk x consequence	Weighted failure	-0.508	[-0.570, -0.446]
Risk x consequence	Task success	1.05 pp	[0.72, 1.38] pp

REFERENCES

1. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," International Conference on Learning Representations (ICLR), 2023. arXiv:2210.03629.
2. X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," International Conference on Learning Representations (ICLR), 2023. arXiv:2203.11171.
3. C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters," arXiv:2408.03314, 2024.
4. N. Lee, L. E. Erdogan, C. J. John, S. Krishnapillai, M. W. Mahoney, K. Keutzer, and A. Gholami, "Agentic Test-Time Scaling for WebAgents," arXiv:2602.12276, 2026.
5. D. Paglieri et al., "Learning When to Plan: Efficiently Allocating Test-Time Compute for LLM Agents," arXiv:2509.03581, 2025.
6. T. Liu et al., "Budget-Aware Tool-Use Enables Effective Agent Scaling," Conference on Language Modeling (COLM), 2026. arXiv:2511.17006.
7. J. Wen, L. He, and Z. He, "Not All Errors Are Equal: Consequence-Aware Reasoning Compute Allocation," arXiv:2606.04402, 2026. DOI: 10.48550/arXiv.2606.04402.
8. Z. Li, J. Huang, X. Guo, G. Wang, and C. Zhang, "Same Signal, Opposite Meaning: Direction-Informed Adaptive Learning for LLM Agents," arXiv:2605.06908, 2026. DOI: 10.48550/arXiv.2605.06908.
9. W. Si, S. Jang, I. Lee, and O. Bastani, "Conformal Constrained Policy Optimization for Cost-Effective LLM Agents," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 40, no. 30, pp. 25446-25453, 2026. DOI: 10.1609/aaai.v40i30.39739.
10. A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, and T. Schuster, "Conformal Risk Control," International Conference on Learning Representations (ICLR), 2024.
11. S. Sadhuka et al., "E-valuator: Reliable Agent Verifiers with Sequential Hypothesis Testing," arXiv:2512.03109, 2025, rev. 2026. DOI: 10.48550/arXiv.2512.03109.
12. D. Prinster et al., "Conformal Policy Control," arXiv:2603.02196, 2026. DOI: 10.48550/arXiv.2603.02196.
13. J. Opoku and D. Banahene, "ToolChain-CRC: Conformal Risk Control for Agentic AI Under Retrieval and Tool-Use Drift," arXiv:2606.18467, 2026. DOI: 10.48550/arXiv.2606.18467.
14. Z. Chen et al., "Cordon: Semantic Transactions for Tool-Using LLM Agents," arXiv:2606.17573, 2026. DOI: 10.48550/arXiv.2606.17573.
15. C. Wu et al., "Safety Does Not Compose: Non-Decaying Loop State for Autonomous LLM Agents," arXiv:2608.27141, 2026. DOI: 10.48550/arXiv.2608.27141.
16. S. Zhou et al., "WebArena: A Realistic Web Environment for Building Autonomous Agents,"

International Conference on Learning Representations (ICLR), 2024. arXiv:2307.13854.
17. B.-W. Kwak et al., "ToolHaystack: Stress-Testing Tool-Augmented Language Models in Realistic Long-Term Interactions," Findings of the Association for Computational Linguistics: EMNLP 2025, 2025. DOI: 10.18653/v1/2025.findings-emnlp.1344.
18. Y. Feng et al., "LongCLI-Bench: A Preliminary Benchmark and Study for Long-horizon Agentic Programming in Command-Line Interfaces," Findings of the Association for Computational Linguistics: ACL 2026, 2026. DOI: 10.18653/v1/2026.findings-acl.1497.
19. T. Kuntz et al., "OS-Harm: A Benchmark for Measuring Safety of Computer Use Agents," Advances in Neural Information Processing Systems 38, Datasets and Benchmarks Track, 2025. DOI: 10.52202/085713-1502.
20. J. Yang, S. Shao, D. Liu, and J. Shao, "RiOSWorld: Benchmarking the Risk of Multimodal Computer-Use Agents," Advances in Neural Information Processing Systems 38, 2025. DOI: 10.52202/085713-0286.
21. T. Sadhu, Y. Chen, and A. Pesaranghader, "VestaBench: An Embodied Benchmark for Safe Long-Horizon Planning Under Multi-Constraint and Adversarial Settings," Proceedings of EMNLP 2025: Industry Track, 2025. DOI: 10.18653/v1/2025.emnlp-industry.149.
22. P. S. Ponduru, "AgentMesh-MCP: A Secure and Governed Framework for Agentic AI Systems Using LLM Agents and Model Context Protocol Servers," International Journal of Scientific Research in Engineering and Management, 2026. DOI: 10.55041/IJSREM62689.
23. J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Advances in Neural Information Processing Systems 35, 2022. arXiv:2201.11903.
24. B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Re, and A. Mirhoseini, "Large Language Monkeys: Scaling Inference Compute with Repeated Sampling," arXiv:2407.21787, 2024.
25. S. Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," Advances in Neural Information Processing Systems 36, 2023. arXiv:2305.10601.
26. M. Besta et al., "Graph of Thoughts: Solving Elaborate Problems with Large Language Models," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 16, pp. 17682-17690, 2024. DOI: 10.1609/aaai.v38i16.29720.
27. A. Madaan et al., "Self-Refine: Iterative Refinement with Self-Feedback," Advances in Neural Information Processing Systems 36, 2023. DOI: 10.52202/075280-2019.
28. N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," Advances in Neural Information Processing Systems 36, 2023. arXiv:2303.11366.
29. S. Hao, Y. Gu, H. Ma, J. Hong, Z. Wang, D. Wang, and Z. Hu, "Reasoning with Language Model is Planning with World Model," Proceedings of EMNLP 2023, pp. 8154-8173, 2023. DOI: 10.18653/v1/2023.emnlp-main.507.
30. A. Zhou, K. Yan, M. Shlapentokh-Rothman, H. Wang, and Y.-X. Wang, "Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models," Proceedings of the 41st International Conference on Machine Learning, PMLR 235, pp. 62138-62160, 2024.
31. K. Cobbe et al., "Training Verifiers to Solve Math Word Problems," arXiv:2110.14168, 2021.
32. H. Lightman et al., "Let's Verify Step by Step," arXiv:2305.20050, 2023.
33. T. Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," Advances in Neural Information Processing Systems 36, 2023. DOI: 10.52202/075280-2997.
34. E. Karpas et al., "MRKL Systems: A Modular, Neuro-Symbolic Architecture That Combines Large Language Models, External Knowledge Sources and Discrete Reasoning," arXiv:2205.00445, 2022.
35. Y. Qin et al., "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs," International Conference on Learning Representations (ICLR), 2024. arXiv:2307.16789.
36. M. Li et al., "API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs," Proceedings of EMNLP 2023, pp. 3102-3116, 2023. DOI: 10.18653/v1/2023.emnlp-main.187.
37. Y. Song et al., "RestGPT: Connecting Large Language Models with Real-World RESTful APIs," arXiv:2306.06624, 2023.
38. S. G. Patil et al., "Gorilla: Large Language Model Connected with Massive APIs," arXiv:2305.15334, 2023.
39. S. Hao, T. Liu, Z. Wang, and Z. Hu, "ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings," Advances in Neural Information Processing Systems 36, 2023. DOI: 10.52202/075280-1988.
40. P. Lu et al., "Chameleon: Plug-and-Play Compositional Reasoning with Large Language

Models," Advances in Neural Information Processing Systems 36, 2023. DOI: 10.52202/075280-1882.
41. Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang, "ToolQA: A Dataset for LLM Question Answering with External Tools," Advances in Neural Information Processing Systems 36, 2023. DOI: 10.52202/075280-2180.
42. S. Yao et al., "WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents," Advances in Neural Information Processing Systems 35, 2022. DOI: 10.52202/068431-1508.
43. X. Liu et al., "AgentBench: Evaluating LLMs as Agents," International Conference on Learning Representations (ICLR), 2024. arXiv:2308.03688.
44. T. Xie et al., "OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments," Advances in Neural Information Processing Systems 37, Datasets and Benchmarks Track, 2024.
45. A. Drouin et al., "WorkArena: How Capable are Web Agents at Solving Common Knowledge Work Tasks?" Proceedings of the 41st International Conference on Machine Learning, PMLR 235, pp. 11642-11662, 2024.
46. J. Y. Koh et al., "VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks," Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, pp. 881-905, 2024. DOI: 10.18653/v1/2024.acl-long.50.
47. C. Ma et al., "AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents," Advances in Neural Information Processing Systems 37, Datasets and Benchmarks Track, 2024. DOI: 10.52202/079017-2365.
48. X. Deng et al., "Mind2Web: Towards a Generalist Agent for the Web," arXiv:2306.06070, 2023.
49. C. E. Jimenez et al., "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" International Conference on Learning Representations (ICLR), 2024. arXiv:2310.06770.
50. J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press, "SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering," Advances in Neural Information Processing Systems 37, 2024. arXiv:2405.15793.
51. R. Wang et al., "ScienceWorld: Is your Agent Smarter than a 5th Grader?" Proceedings of EMNLP 2022, 2022. DOI: 10.18653/v1/2022.emnlp-main.775.
52. Y. Ruan et al., "ToolEmu: Identifying the Risks of LM Agents with an LM-Emulated Sandbox," arXiv:2309.15817, 2023.
53. E. Debenedetti et al., "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents," Advances in Neural Information Processing Systems 37, Datasets and Benchmarks Track, 2024. DOI: 10.52202/079017-2636.
54. Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents," Findings of the Association for Computational Linguistics: ACL 2024, pp. 10471-10506, 2024. DOI: 10.18653/v1/2024.findings-acl.624.
55. J. Ye et al., "ToolSword: Unveiling Safety Issues of Large Language Models in Tool Learning Across Three Stages," Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024. DOI: 10.18653/v1/2024.acl-long.119.
56. K. Greshake et al., "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," arXiv:2302.12173, 2023.
57. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," Proceedings of the 34th International Conference on Machine Learning, PMLR 70, pp. 1321-1330, 2017.
58. S. Kadavath et al., "Language Models (Mostly) Know What They Know," arXiv:2207.05221, 2022.
59. L. Kuhn, Y. Gal, and S. Farquhar, "Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation," International Conference on Learning Representations (ICLR), 2023.
60. S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting Hallucinations in Large Language Models Using Semantic Entropy," Nature, vol. 630, pp. 625-630, 2024. DOI: 10.1038/s41586-024-07421-0.
61. P. Manakul, A. Liusie, and M. J. F. Gales, "SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models," Proceedings of EMNLP 2023, 2023. DOI: 10.18653/v1/2023.emnlp-main.557.
62. Y. Geifman and R. El-Yaniv, "Selective Classification for Deep Neural Networks," Advances in Neural Information Processing Systems 30, 2017.
63. Y. Geifman and R. El-Yaniv, "SelectiveNet: A Deep Neural Network with an Integrated Reject Option," Proceedings of the 36th International Conference on Machine Learning, PMLR 97, pp. 2151-2159, 2019.
64. A. N. Angelopoulos and S. Bates, "A Gentle Introduction to Conformal Prediction and

Distribution-Free Uncertainty Quantification," arXiv:2107.07511, 2021.
65. I. Gibbs and E. Candes, "Adaptive Conformal Inference Under Distribution Shift," Advances in Neural Information Processing Systems 34, 2021.
66. R. F. Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani, "Conformal Prediction Beyond Exchangeability," Annals of Statistics, vol. 51, no. 2, 2023. arXiv:2202.13415.
67. A. N. Angelopoulos, S. Bates, E. J. Candes, M. I. Jordan, and L. Lei, "Learn then Test: Calibrating Predictive Algorithms to Achieve Risk Control," Annals of Applied Statistics, 2025. arXiv:2110.01052.
68. S. Bates, A. N. Angelopoulos, L. Lei, J. Malik, and M. I. Jordan, "Distribution-Free, Risk-Controlling Prediction Sets," Journal of the ACM, vol. 68, no. 6, pp. 1-34, 2021.
69. R. J. Tibshirani, R. F. Barber, E. J. Candes, and A. Ramdas, "Conformal Prediction Under Covariate Shift," arXiv:1904.06019, 2019.
70. M. Campos, A. Farinhas, C. Zerva, M. A. T. Figueiredo, and A. F. T. Martins, "Conformal Prediction for Natural Language Processing: A Survey," Transactions of the Association for Computational Linguistics, vol. 12, pp. 1497-1516, 2024. DOI: 10.1162/tacl_a_00715.
71. J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained Policy Optimization," Proceedings of the 34th International Conference on Machine Learning, PMLR 70, pp. 22-31, 2017.
72. J. Garcia and F. Fernandez, "A Comprehensive Survey on Safe Reinforcement Learning," Journal of Machine Learning Research, vol. 16, no. 42, pp. 1437-1480, 2015.
73. M. Alshiekh et al., "Safe Reinforcement Learning via Shielding," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018. DOI: 10.1609/aaai.v32i1.11797.
74. Y. Chow, A. Tamar, S. Mannor, and M. Pavone, "Risk-Sensitive and Robust Decision-Making: A CVaR Optimization Approach," Advances in Neural Information Processing Systems 28, 2015.
75. E. Altman, Constrained Markov Decision Processes. Boca Raton, FL, USA: Chapman & Hall/CRC, 1999. ISBN: 9780849303821.
76. G. Dulac-Arnold et al., "Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis," Machine Learning, vol. 110, pp. 2419-2468, 2021. DOI: 10.1007/s10994-021-05961-4.
77. D. B. Rubin, "Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies," Journal of Educational Psychology, vol. 66, no. 5, pp. 688-701, 1974. DOI: 10.1037/h0037350.
78. P. W. Holland, "Statistics and Causal Inference," Journal of the American Statistical Association, vol. 81, no. 396, pp. 945-960, 1986. DOI: 10.1080/01621459.1986.10478354.
79. P. R. Rosenbaum and D. B. Rubin, "The Central Role of the Propensity Score in Observational Studies for Causal Effects," Biometrika, vol. 70, no. 1, pp. 41-55, 1983. DOI: 10.1093/biomet/70.1.41.
80. B. Efron, "Bootstrap Methods: Another Look at the Jackknife," Annals of Statistics, vol. 7, no. 1, pp. 1-26, 1979. DOI: 10.1214/aos/1176344552.
81. F. Wilcoxon, "Individual Comparisons by Ranking Methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80-83, 1945. DOI: 10.2307/3001968.
82. S. Holm, "A Simple Sequentially Rejective Multiple Test Procedure," Scandinavian Journal of Statistics, vol. 6, no. 2, pp. 65-70, 1979.
83. Q. McNemar, "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages," Psychometrika, vol. 12, no. 2, pp. 153-157, 1947. DOI: 10.1007/BF02295996.
84. R. D. Wright and J. B. Ramsay, "On the Effectiveness of Common Random Numbers," Management Science, vol. 25, no. 7, pp. 649-656, 1979. DOI: 10.1287/mnsc.25.7.649.
85. J. P. C. Kleijnen, "Analyzing Simulation Experiments with Common Random Numbers," Management Science, vol. 34, no. 1, pp. 65-74, 1988. DOI: 10.1287/mnsc.34.1.65.