
Multi-Turn Optimisation for Runtime Monitoring

Yuanyuan Sun
Open Community for AI Safety
China

Yuting Wu
China Telecom

With
Apart Research

Abstract

We study whether session history helps allocate runtime-monitoring budget in multi-turn interactions. Building on *Combining Cost-Constrained Runtime Monitors for AI Safety*, we compare single-monitor baselines, a static two-monitor threshold baseline, and a flexible session-aware baseline on SafeDialBench English conversations. Because the original monitor runner was not available, we use proxy alpha and beta scores anchored to the parent repository's monitor-score distributions.

Our main result is that the session-aware baseline appears to improve the budget-recall tradeoff in this proxy setup. On one held-out split, it matches the static baseline's best attack-turn recall (0.763) at a much lower total budget (2.840 vs 5.212). Across 10 random session splits, mean best recall remains nearly identical for the session-aware and static baselines (0.822 vs 0.823), while mean best total budget is lower for the session-aware baseline (4.090 vs 5.770). This is a multi-turn runtime-monitoring study inspired by AI control, not a full control-faithful evaluation.

1. Introduction

This project builds on *Combining Cost-Constrained Runtime Monitors for AI Safety* (Hua et al., 2025), which studies how to combine multiple runtime monitors under a unified budget constraint.

That paper gives the basic setting for this project: a cheap monitor, a stronger expensive monitor, and a limited budget that also has to account for human audit. Our work extends that setup to the multi-turn case.

We study a practical runtime-monitoring question: **how much does multi-turn session history improve the decision of when to invoke a stronger monitor?** This matters because many risky interactions emerge gradually across several turns. If session history adds useful information, then companies may be able to achieve the same safety at lower cost, or higher safety at the same cost. The project is motivated by AI control, but the present evaluation is narrower: it focuses on budgeted runtime-monitor routing in multi-turn interactions rather than a full control protocol with catastrophic-failure semantics.

Our main contributions are:

1. We build a multi-turn monitoring pipeline on top of SafeDialBench-format conversations.
2. We compare routing policies, including the static two-monitor threshold baseline inherited from Hua et al. (2025) and a flexible session-aware baseline.
3. We run a held-out train/test evaluation suggesting that session-aware routing can improve the budget-recall tradeoff in our proxy setup.

2. Related Work

This work is closest to:

- **Combining cost-constrained runtime monitors**, which provides the baseline and unified-budget framing that this project extends. See Hua et al. (2025): <https://arxiv.org/pdf/2507.15886>
- **Multi-turn LLM safety benchmarks**, especially SafeDialBench, because the main extension in this project is from single-turn routing to multi-turn routing.

Relative to Hua et al. (2025), the practical change in this project is that we test whether session history is useful for monitor routing in a multi-turn conversational setting. Relative to the broader AI control literature, this project should be understood as a stepping stone: it investigates multi-turn monitor allocation in a proxy conversational benchmark, rather than end-to-end control on a control-faithful dataset.

3. Methods

We used a public SafeDialBench English JSONL dataset and converted it into turn-level monitoring rows while preserving session structure. We chose SafeDialBench English because it is public, safety-focused, explicitly multi-turn, and small enough to use within the project timeframe while still providing realistic conversational escalation patterns. It is not an ideal AI-control dataset: it is a conversational safety benchmark rather than a control-faithful coding/backdoor benchmark with catastrophic-failure semantics. However, within the available time we did not have a practical public multi-turn control/backdoor dataset plus a runnable matching monitor pipeline, so SafeDialBench

was the most workable option for testing the multi-turn routing idea itself. The final cleaned dataset contained:

- 346 conversations
- 1673 assistant turns

We did not use the original monitor-score pipeline because the repository provides processed monitor-score CSVs, but not a turnkey end-to-end scorer for new arbitrary conversation turns. Reproducing real alpha/beta scores on SafeDialBench would likely have required recovering the original `inspect_ai` monitor tasks, model-serving setup, and raw logging pipeline, then rerunning the monitors over the new multi-turn dataset. Given the project timeframe, this reconstruction was too uncertain and time-intensive, so we used a proxy scoring setup. Each turn was mapped to:

- `beta_score`: a cheap weaker monitor
- `alpha_score`: a stronger expensive monitor

These proxy scores were anchored to the repository's existing BCB monitor-score distributions. We chose this proxy design for a practical reason: it preserves the rough scale and separation behavior of the repository's existing alpha/beta monitors, while still letting us score new multi-turn conversation data.

Turn labels were also proxy labels. We derived them from dataset metadata and simple heuristics over task/method/scene because SafeDialBench does not provide monitor-specific turn-level success labels. This is a limitation, but it gives a usable first-pass signal for comparing routing policies.

So, concretely, the SafeDialBench experiment is testing the following question: **given multi-turn conversational risk signals, does session history help decide when to escalate from a cheap monitor to a stronger monitor and then to human review?** SafeDialBench does not model self-jailbreaking or model-to-monitor deception directly; instead, it provides multi-turn adversarial conversational pressure, which we use as a proxy environment for studying session-aware monitor routing. It is therefore not testing whether the monitors detect true control-style catastrophic failures on a control-faithful benchmark.

We compared four policies:

- **Beta only**: use only the cheap monitor and audit above a threshold.
- **Alpha only**: use only the strong monitor and audit above a threshold.
- **Baseline two-monitor region**: use beta first, then call alpha when beta crosses a fixed threshold. This is the closest multi-turn analogue of the static two-monitor baseline from Hua et al. (2025).
- **Flexible session baseline**: use beta first, then decide whether to call alpha using current-turn information plus summary features from earlier turns in the same conversation.

In all four cases, human audit is used as the final intervention step in the main experiment. Each policy first produces a final risk score for the turn. That score is then compared with an audit threshold. If it crosses the threshold, the turn is sent to human review; if it does not, the turn is not counted as caught. The policies differ only in how they produce that final score:

- **Beta only** audits from the beta posterior alone.
- **Alpha only** audits from the alpha posterior alone.
- **Baseline two-monitor region** audits from beta alone unless beta first decides to call alpha, in which case it audits from the combined alpha+beta posterior.
- **Flexible session baseline** uses session history to decide when to call alpha, then audits from either beta alone or the combined alpha+beta posterior.

The session-aware flexible baseline uses simple history features such as:

- `turn_idx`
- `beta_score`
- `prior_before_turn`
- `beta_posterior`
- `prev_beta_score`
- `prev_beta_mean`
- `prev_beta_max`
- `prev_beta_post_mean`
- `beta_delta`
- `num_prev_high_beta`
- `num_prev_high_post`
- `task`

The flexible gate was trained on a routing-aligned target: whether calling alpha would improve the final decision at that turn, rather than simply whether the turn looked harmful.

Following the paper's unified-budget framing, we account for both monitor calls and audit cost. In our proxy setup we used:

- beta monitor cost: 0.21
- alpha monitor cost: 1.0
- audit cost: 20.0

The main accounting equation is:

$$total\ budget = monitor\ budget + benign\ audit\ rate * c_a$$

where $c_a = 20$.

For policies that start with beta, monitor budget is:

$$monitor\ budget = c_n + call\ alpha * c_m$$

where $c_n = 0.21$ and $c_m = 0.21$.

The flexible gate itself is a logistic regression model trained to predict whether calling alpha helps:

$$p(call\ alpha\ helps | x) = \sigma(w^T x + b)$$

and alpha is called when:

$$p(\text{call alpha helps} \mid x) \geq \tau_{\text{gate}}$$

where (x) is the session feature vector and τ_{gate} is selected during threshold search.

To make evaluation honest, we split sessions into train and test. Here, "held-out test" means conversations that were not used to choose thresholds or tune the policy:

1. tune policy thresholds on train only
2. for each budget, select the best train-feasible setting
3. report attack-turn recall on held-out test only

4. Results

Our main figure is the 10-split stability curve:

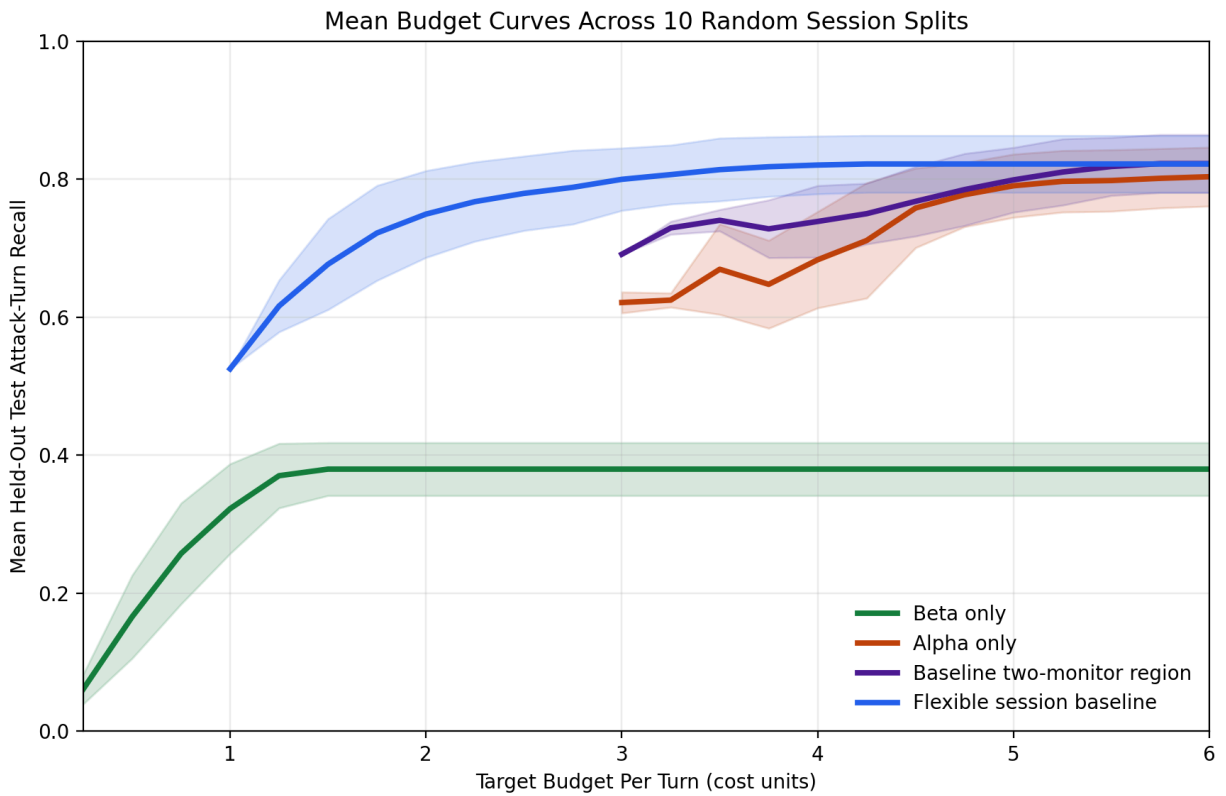


Figure 1. Mean budget curves across 10 random train/test session splits. The x-axis is **target budget per turn**, measured in the same cost units used throughout the project ($\text{beta} = 0.21$, $\text{alpha} = 1.0$, $\text{audit} = 20.0$). The y-axis is **mean held-out test attack-turn recall**, i.e. the average fraction of harmful turns caught on the held-out test split. The solid line shows the mean recall across splits at each budget, and the shaded band shows vertical variation across splits.

Higher is better on the y-axis; further left is cheaper on the x-axis. Flat segments should be read as saturation on the evaluated threshold grid: once a policy has already reached its best real operating point under a given budget, allowing more budget does not change the plotted value until another distinct operating point becomes available.

The corresponding summary data is: `multi_turn_split_stability_curve_stats.csv`

The main result is that the flexible session baseline appears more budget-efficient than the static two-monitor baseline in this proxy setup.

The main observation is:

- `Beta only` is cheapest but weak, which is expected because it never uses the stronger monitor.
- `Alpha only` is stronger but expensive, which is also expected because it always pays for the strong monitor.
- `Baseline two-monitor region` reaches high recall, but only at high budget.
- `Flexible session baseline` matches the best held-out recall of the static baseline at much lower budget in this proxy experiment.

Relative to the baseline inherited from Hua et al. (2025), this suggests that conversation history may provide useful routing information beyond the current turn's score alone.

Quantitatively, the main effect size is now best understood from the 10-split aggregate:

- static baseline mean best held-out recall: `0.823` at mean best budget `5.770`
- flexible session baseline mean best held-out recall: `0.822` at mean best budget `4.090`

So across 10 random session splits in this proxy setup, the multi-turn routing policy reaches nearly the same best recall as the static baseline at about 29% lower mean total budget.

The figure should not be interpreted as meaning that each strategy can only exist in one narrow budget region in principle. Instead, each strategy spans only part of the x-axis because we plot real threshold-sweep operating points, and many nearby thresholds collapse to the same realized spend on this finite dataset. A flat line segment means that, over that budget range, the best feasible real operating point has not changed yet. This is especially visible for `Beta only`: it has only one monitor and only one real decision knob, the audit threshold, so after a few threshold settings it runs out of distinct useful operating points and the curve stays flat. The same logic applies to the other curves as well.

More concretely:

- `Beta only` only changes through the audit threshold, so it saturates quickly. Once the best beta-only point under budget has been reached, giving it more budget does not help unless another distinct audit threshold produces a different real outcome.
- `Alpha only` always pays the strong-monitor cost, so it cannot appear at very low budget. It also stays flat across some budget ranges because its total cost is driven heavily by the audit

threshold, and many nearby thresholds produce the same realized audit rate on a finite dataset.

- `Baseline two-monitor region` and `Flexible session baseline` have wider ranges because they can change both alpha usage and audit usage, but they still only occupy part of the x-axis because some budget levels simply have no new real operating point available in the sweep.

5. Discussion and Limitations

The main implication is that multi-turn routing appears practically useful. The benefit does not seem to come from a complicated new theory; it comes from extending the paper's cost-constrained monitor-combination idea with memory of earlier turns. In this setup, remembering how risky the conversation has been so far seems to help decide when the strong monitor is worth invoking.

Another practical implication is that the model exposes explicit thresholds and cost-sensitive tradeoffs. In a production system, this is useful because companies may want to tune the routing behavior based on business needs. For example, a company with strict latency constraints may choose a higher gate threshold and lower alpha usage, while a company in a higher-risk domain may choose a lower gate threshold and spend more on monitoring and audit.

The main limitations are:

- monitor scores are proxy scores, not outputs from the original monitor runner
- turn labels are heuristic, not gold production labels
- session features are still simple summary statistics rather than deep semantic history
- the cost model is approximate rather than production-complete
- the dataset is a compromise: `SafeDialBench` gives real multi-turn structure, but it is not the most appropriate dataset for AI-control-style evaluation because it is conversational safety data rather than control-faithful coding/backdoor data

This last point is important for interpreting the project honestly. We realized partway through the work that a more relevant control/backdoor-style multi-turn dataset would be preferable, but we were not able to obtain or construct one, together with a matching scoring pipeline, within the remaining hackathon time. As a result, we kept `SafeDialBench` and narrowed the claim accordingly.

So these results should be read as evidence that multi-turn optimisation is plausible and promising, not as a final deployment claim. More specifically, they support a stepping-stone claim: session history seems useful for budgeted runtime-monitor routing, which is one component of a broader AI control stack, but this project does not yet evaluate that claim on a control-faithful benchmark with real monitor outputs.

As a stronger robustness check, we reran the full train/test experiment across 10 random session splits and summarized the mean budget curve and the distribution of each policy's best held-out operating point. The main qualitative conclusion stayed the same: the flexible session baseline remained more budget-efficient than the static baseline, while their best recall stayed nearly identical on average.

Across 10 splits:

- mean best recall for the flexible session baseline: 0.822
- mean best recall for the static baseline: 0.823
- mean best budget for the flexible session baseline: 4.090
- mean best budget for the static baseline: 5.770

So the flexible policy remains cheaper on average while matching the static baseline's best recall up to a very small difference in mean performance. The split-to-split variation is still meaningful, so this should be treated as a moderate-stability result rather than a final high-confidence deployment claim. This robustness pattern is already visible in Figure 1.

Natural next steps are:

- build or obtain a control-faithful multi-turn dataset, ideally based on public coding/backdoor tasks (for example APPS or BigCodeBench-style tasks decomposed into multi-step trajectories)
- define honest and subversive trajectories on those tasks, with both turn-level and sequence-level labels
- run a real cheap/strong monitor pair on each step of those trajectories instead of proxy scoring
- evaluate the same routing framework on logged agent trajectories rather than dialogue-only benchmarks, for example trajectory-level safety datasets such as ATBench
- replace proxy scores with real monitor outputs on a multi-turn dataset
- improve session features, especially escalation-related features
- evaluate additional session-level metrics such as time-to-first-catch
- explore whether a stronger gate model or richer session features improve the train/test curve
- expose the trained thresholds and budget-recall tradeoff more directly as deployment knobs for different operational settings

6. Conclusion

We studied whether multi-turn context helps decide when to invoke a stronger runtime monitor. Building on *Combining Cost-Constrained Runtime Monitors for AI Safety*, we adapted its budgeted two-monitor framing to a conversational setting. Using a SafeDialBench-based proxy setup and a held-out train/test evaluation, we found evidence that a flexible session-aware baseline can improve the budget-recall tradeoff relative to simple single-monitor baselines and can match a static two-monitor baseline at lower budget.

The result is still preliminary because the current setup relies on proxy monitor scores and heuristic labels, and because the dataset is not fully control-faithful. Even so, the overall direction looks useful: multi-turn optimisation appears to be a practical way to make runtime monitoring more cost-effective, and it provides a concrete starting point for future evaluations on stronger AI-control-style data.

Code and Data

Code repository: <https://github.com/YuanyuanSunSarah/apart-2026-multi-turn-runtime-monitoring>

Main report artifact:

- `multi_turn_optimisation_report.md`

Main figure:

- `multi_turn_split_stability_mean_curve.png`

Data / datasets:

- SafeDialBench public dataset: <https://github.com/drivetosouth/SafeDialBench-Dataset>
- Local cleaned file used in this project: `safedialbench_full_valid.jsonl`

Other artifacts:

- Single-split held-out curve: `multi_turn_budget_curve.png`
- No-human-audit comparison: `multi_turn_no_human_budget_curve.png`
- 10-split stability summary: `multi_turn_split_stability_aggregate.csv`

References

Hua, Y. et al. (2025). *Combining Cost-Constrained Runtime Monitors for AI Safety*. arXiv. <https://arxiv.org/pdf/2507.15886>

SafeDialBench authors. (2025). *SafeDialBench*. arXiv / project repository. <https://arxiv.org/abs/2502.11090> and <https://github.com/drivetosouth/SafeDialBench-Dataset>

Appendix

Supplementary material produced in this project includes:

- full held-out train/test metrics: `multi_turn_train_metrics.csv`
`multi_turn_test_metrics.csv`
- 10-split stability summaries: `multi_turn_split_stability_summary.csv`
`multi_turn_split_stability_aggregate.csv`
- split-stability mean curve: `multi_turn_split_stability_mean_curve.png`

LLM Usage Statement

LLM assistance was used during the development of the project code, plotting, and report drafting. All reported metrics, figures, and claims were checked against files generated in the repository, and the final conclusions were manually reviewed.