DOES EXECUTION ORDER IMPLY CAUSALITY? FEDERATED POST-TRAINING OF ASYNCHRONOUS LLM AGENTS OVER EXECUTION POSETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language model agents often execute multiple tools and subtasks concurrently, while the resulting traces remain distributed across private clients. Federated adaptation allows these clients to improve a shared agent without centralizing their execution data. However, existing methods usually convert each execution into a single token sequence. In asynchronous environments, independent events may finish in different orders because of tool latency, queueing, or local scheduling. As a result, two clients can complete the same execution, observe the same tool outputs, and make the same decision, yet store the intermediate events in different orders. Sequential training treats these order differences as different positions and contexts, which produces different local gradients from the same causal execution. We call this problem *interleaving heterogeneity*. To address it, we propose TopoFed-Agent, which represents each execution as a partial order certified by runtime evidence. During local adaptation, each event attends only to its causal ancestors, uses a position determined by its causal rank, and receives supervision attached to the event itself. In this way, changing the legal storage order of independent events does not change the learning problem. We prove that, for a fixed certified execution, all legal serializations produce identical predictions, losses, and parameter gradients. Therefore, gradient differences caused only by client schedulers are eliminated. Experiments across four execution structures and five Transformer initializations show that sequential training yields relative full-parameter gradient deviations ranging from 0.26 to 0.46 across legal serializations, whereas TopoFed-Agent keeps the deviation below 2.31×10^{-16}. The code for this work can be obtained from this supplementary file.

1 INTRODUCTION

Language model agents are increasingly used to complete tasks that involve multiple tools, applications, and other agents (Qin et al., 2024). A coding agent may search a repository while running tests, and a service agent may query inventory and payment systems at the same time (Wang et al., 2024). Such executions generate rich interaction traces, but these traces are often distributed across private users and organizations. Federated adaptation provides a practical way to learn from them without centralizing local execution data. Recent studies have also extended federated learning to language models and interactive agents (Bai et al., 2024; Chen et al., 2026a).

Most existing methods treat each local execution as one ordered training sequence (Qi et al., 2025). In asynchronous workflows, however, one execution may have several valid orders. For example, an agent may launch a repository search and a test command concurrently. One client may receive the search result first, while another receives the test result first because of different tool latency or queueing. Both clients observe the same two results and make the same later decision. Their executions are therefore causally equivalent, but the stored transcripts are different. Sequential training assigns the two results different token positions and different preceding contexts. The same execution can then produce different predictions and gradients. Thus, it raises a fundamental question: ***Should one execution produce different updates only because its independent events are stored in different legal orders?***

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.

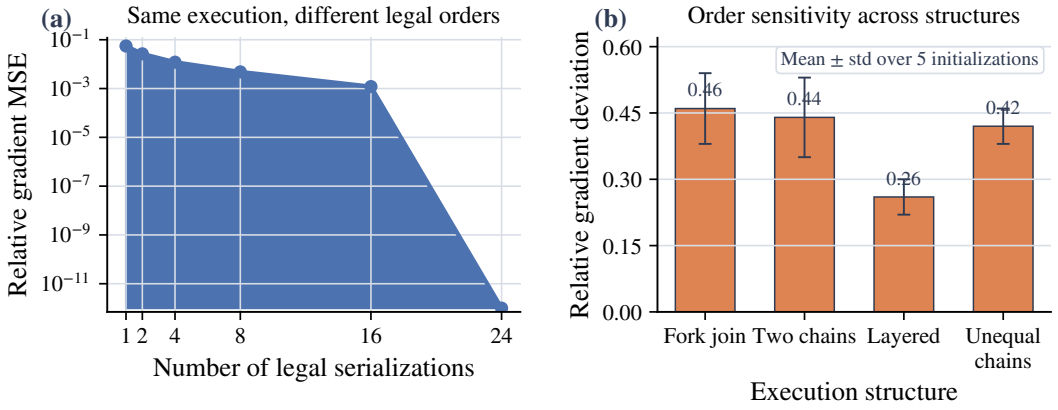


Figure 1: Motivation of interleaving heterogeneity under controlled synthetic executions.

To examine whether serialization order alone can change the learning signal, we construct controlled synthetic executions in which the execution graph, event contents, prediction targets, and model parameters are fixed. We use four execution structures: fork join, two chains, layered, and unequal chains. The experiments use a two layer Transformer with two attention heads, a hidden size of 16, and a feed forward size of 32. We evaluate five random initializations in CPU float64 with dropout disabled. The model is not trained. For each execution, we vary only the legal serialization of independent events and compare the resulting full parameter gradients. Based on these settings, we observe the following two phenomena.

(1) One execution can produce many different gradient observations (Figure 1(a)). We begin with a four branch fork join execution, which admits 24 legal serializations. When only one serialization is used, the relative mean square error of its gradient with respect to the average over all legal orders is 5.53×10^{-2}. Averaging over 2, 4, 8, and 16 legal serializations reduces this error to 2.64×10^{-2}, 1.20×10^{-2}, 4.81×10^{-3}, and 1.20×10^{-3}, respectively. The error reaches numerical zero only when all 24 legal orders are included. The execution itself never changes. The variation is introduced only by how independent events are serialized.

(2) The same problem appears across different execution structures (Figure 1(b)). We then repeat the same controlled comparison on four execution structures: fork join, two chains, layered, and unequal chains. Sequential serialization produces mean full parameter gradient deviations of 0.46, 0.44, 0.26, and 0.42, respectively, over five random initializations. In every case, the execution graph, event contents, and event level targets remain fixed. Only the legal order changes. These results show that serialization order alone can alter the learning signal, and the effect is not specific to one execution structure.

These observations reveal a source of client disagreement beyond conventional data or task heterogeneity. We call it *interleaving heterogeneity*: clients with the same causal execution and supervision may obtain different gradients solely because independent events are serialized differently. This discrepancy arises when a partially ordered execution is forced into a single sequence and therefore cannot be removed by downstream aggregation. Sampling multiple legal orders only reduces the effect at additional cost, while timestamps still reflect runtime scheduling rather than causal dependence. Since a single ordered transcript cannot distinguish causality from incidental completion order, the execution structure must be represented explicitly.

In light of these observations, we propose TopoFed-Agent for federated adaptation over execution partial orders. Each client uses runtime evidence to construct a conservative execution graph. Confirmed dependencies are retained. Independent events are separated only when their independence can be established from the execution. The graph is then converted into tokens for local adaptation. Each token attends only to its causal ancestors. Its position is determined by causal rank instead of its location in the stored sequence. Supervision is attached to the corresponding event. Changing the legal storage order of independent events therefore does not change the context, position, or target used for learning. The client can then use the original optimizer and return an ordinary model update to the server. The main contributions of this work are as follows.

- We identify and define *interleaving heterogeneity* in federated agent adaptation. We show that different legal orders of the same asynchronous execution can change local predictions and gradients even when the event contents and prediction targets remain unchanged.
- We propose TopoFed-Agent, which represents local executions as runtime certified partial orders and uses ancestor attention, causal rank positions, and event local supervision to remove the effect of legal storage order from local adaptation.
- Experiments across four execution structures and five Transformer initializations verify the result under different order budgets, schedulers, client scales, concurrency widths, and incomplete provenance.

2 Related Work

Federated language model adaptation has been widely studied to reduce communication and memory costs and to support heterogeneous clients through efficient updates and parameter allocation (Qin et al., 2024; Wang et al., 2024; Bai et al., 2024; Panchal et al., 2024; Wu et al., 2024; Zheng et al., 2024; Liu et al., 2024). Recent work has further extended federated learning to interactive agents, where clients may differ in tasks, environments, world models, or routing behaviors (Chen et al., 2026a; Hou, 2026; Chakraborty et al., 2026). Meanwhile, agent trajectory optimization improves exploration and credit assignment through sequential interaction, process feedback, transition level supervision, and branched trajectory structures (Qi et al., 2025; Song et al., 2024; Xiong et al., 2024; Su et al., 2025; Liu et al., 2026; Luo et al., 2025; Zhang, 2026; He et al., 2026; Li et al., 2026). These methods mainly optimize what agents learn from a trajectory, but usually assume that the observed trajectory order is itself a valid training order. For asynchronous executions, however, different completion orders may correspond to the same underlying execution. Partial order process mining, DAG based representations, and Hasse diagram models provide tools for representing concurrency without committing to one arbitrary topological order (Leemans et al., 2023; Li & Guo, 2026; Dong et al., 2022; Thost & Chen, 2021), while runtime tracing systems provide event identities and dependency evidence that can help recover such execution relations (OpenTelemetry Authors, 2026; World Wide Web Consortium, 2024; Chen et al., 2026b; Mansoor et al., 2026; AlSayyad et al., 2026). Different from these studies, our work focuses on the learning inconsistency caused by different legal serializations of the same execution in federated agent adaptation, and uses certified execution relations to make local supervision and Transformer gradients invariant to such serialization changes.

3 Problem Definition

3.1 Federated Asynchronous Execution Model

We consider a federated agent system with K clients and a shared policy parameterized by θ. Each client keeps its execution histories local during federated adaptation. Client $k \in [K]$ has aggregation weight $p_k \geq 0$, with $\sum_{k=1}^{K} p_k = 1$.

An asynchronous execution may contain both causally dependent and concurrent events. Since independent events can be recorded in different legal orders, a sequential transcript does not uniquely describe the underlying execution structure. We therefore represent each local execution as an attributed partial order.

Definition 1 (Attributed execution poset)**.** *An attributed execution poset is*

$$G = (V, \prec, \{B_v\}_{v \in V}, D, \{y_v\}_{v \in D}, r), \tag{1}$$

where V is the set of runtime events, $\prec$ is a strict partial order, B_v is the token block of event v, $D \subseteq V$ contains supervised decision events, y_v is the target attached to v, and r is an optional execution level verifier reward.

The relation $u \prec v$ requires u to precede v. If neither $u \prec v$ nor $v \prec u$ holds, the two events are unordered. A legal serialization is any linear extension $\sigma \in \mathrm{LE}(G)$ that preserves $\prec$. The relation is supported by runtime provenance including program order, message send and receive relations, tool call and result links, explicit result consumption, fork and join information, and versioned read and write conflicts. Schedule dependent information such as duration, timeout, or cost remains in B_v

when it affects the agent decision or execution outcome. Two traces are treated as serializations of the same execution only when they preserve the same events, dependencies, decision cuts, targets, and terminal state.

3.2 Interleaving Heterogeneity

Conventional agent adaptation optimizes a sequential loss over one serialization of each execution. Let $\mu_k(G)$ denote the semantic execution distribution of client k, and let $q_k(\sigma \mid G)$ denote its distribution over legal serializations. The corresponding federated objective is

$$F_{\mathrm{seq}}(\theta) = \sum_{k=1}^{K} p_k \mathbb{E}_{G\sim\mu_k} \mathbb{E}_{\sigma\sim q_k(\cdot|G)} \left[\mathcal{L}_{\mathrm{seq}}(\theta; \sigma(G))\right]. \tag{2}$$

Although every $\sigma \in \mathrm{LE}(G)$ preserves the same causal execution, sequential training may assign different positions, contexts, attention patterns, and gradients to different legal serializations. The objective therefore depends not only on the execution distribution μ_k, but also on the serialization distribution q_k. Two clients can consequently obtain different local updates even when their underlying executions and supervision are identical.

Definition 2 (Interleaving heterogeneity)**.** Interleaving heterogeneity *is the discrepancy in client learning signals caused by different legal serializations of the same attributed execution poset. In the scheduler only setting, clients share the same semantic execution distribution $\mu_k = \mu$ but differ in $q_k(\sigma \mid G)$.*

This discrepancy differs from conventional data or task heterogeneity because it arises after the same causal execution is converted into different sequences.

3.3 Schedule Invariant Federated Adaptation Problem

To remove serialization dependent learning differences, the learning unit should be determined by the causal execution rather than one observed sequence. For a supervised decision event $v \in D$, we define

$$\mathcal{L}_{\mathrm{poset}}(\theta; G) = -\sum_{v\in D} \sum_{j=1}^{|y_v|} w_v \log \pi_\theta \left(y_{v,j} \mid \{B_u : u \prec v\}, B_v, y_{v,<j}\right), \tag{3}$$

where $w_v = 1$ for supervised fine tuning and may be a fixed externally computed advantage when a policy gradient surrogate is used. The resulting federated objective is

$$\min_\theta \; F_{\mathrm{poset}}(\theta) = \sum_{k=1}^{K} p_k \mathbb{E}_{G\sim\mu_k} \left[\mathcal{L}_{\mathrm{poset}}(\theta; G)\right]. \tag{4}$$

Unlike Eq.2, this objective contains no client specific serialization distribution. For every fixed execution G and any $\sigma, \sigma' \in \mathrm{LE}(G)$, schedule invariant adaptation further requires

$$\mathcal{L}(\theta; \sigma(G)) = \mathcal{L}(\theta; \sigma'(G)), \tag{5}$$

$$\nabla_\theta \mathcal{L}(\theta; \sigma(G)) = \nabla_\theta \mathcal{L}(\theta; \sigma'(G)). \tag{6}$$

The event contents, execution relations, decision targets, and runtime provenance remain local to each client. The resulting problem is to preserve causal information while removing learning differences introduced only by legal serialization.

4 TopoFed-Agent: Federated Adaptation over Execution Posets

4.1 Overview

We propose TopoFed-Agent to remove learning differences caused by legal serialization without changing the underlying federated training protocol. As shown in Figure. 2, each client constructs a

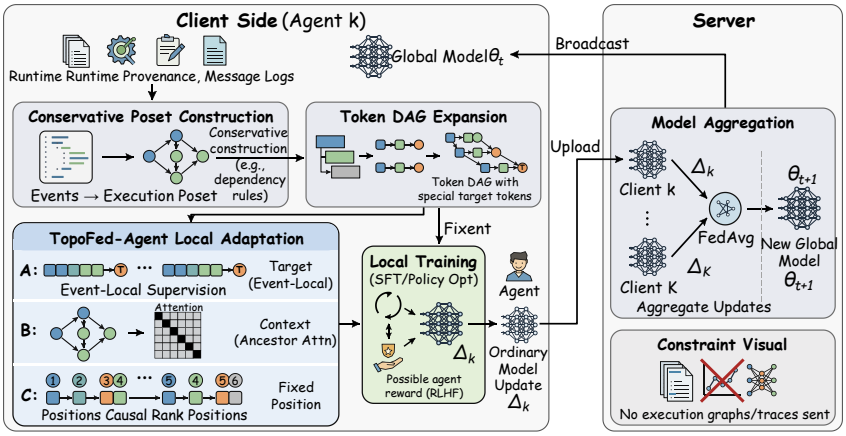


Figure 2: Overview of TopoFed-Agent.

Algorithm 1 TopoFed-Agent at Communication Round t

Require: Global model θ_t; selected clients S_t

1: Server broadcasts θ_t to S_t
2: **for** each client $k \in S_t$ in parallel **do**
3: Construct execution posets G from local runtime provenance
4: Expand each G into token DAG $\widetilde{G}$ with event local supervision
5: Choose any legal packing $\sigma \in \mathrm{LE}(\widetilde{G})$
6: Construct ancestor mask M^{σ} and causal ranks ρ
7: Run the original local SFT or policy optimizer from θ_t
8: Upload model update Δ_k
9: **end for**
10: $\theta_{t+1} \leftarrow \mathsf{Aggregate}\left(\theta_t, \{\Delta_k : k \in S_t\}\right)$

conservative execution poset from runtime provenance and expands it into a token DAG. Event local supervision fixes the prediction target, ancestor attention restricts each token to its causal context, and causal rank positions make positional information independent of tensor order. The resulting representation is processed by the original local optimizer, and only the model update is returned to the server. Algorithm 1 summarizes the local representation, adaptation, and aggregation procedure in each communication round.

4.2 Runtime Certified Poset Construction

Let $\tau = (v_1, \ldots, v_{|V|})$ denote the observed event order. From runtime provenance, the client extracts a set of certified dependency edges $\mathcal{E}^{\mathrm{dep}}$ and a set $\mathcal{I}^{\mathrm{ind}}$ containing event pairs whose independence is explicitly supported. The dependency evidence includes program order, message send and receive relations, tool call and result links, explicit result consumption, and versioned read and write conflicts.

We initialize the execution relation using the certified dependencies,

$$\mathcal{R}^{(0)} = \mathcal{E}^{\mathrm{dep}}. \tag{7}$$

For each pair that remains unordered under $\mathcal{R}^{(0)}$, the observed order is preserved unless independence is certified:

$$\mathcal{R} = \mathcal{E}^{\mathrm{dep}} \cup \left\{(u, v) : u \lhd_{\tau} v,\ u \parallel_{\mathcal{E}^{\mathrm{dep}}} v,\ (u, v) \notin \mathcal{I}^{\mathrm{ind}}\right\}, \tag{8}$$

where $u \lhd_{\tau} v$ means that u appears before v in the observed trace, and $u \parallel_{\mathcal{E}^{\mathrm{dep}}} v$ means that neither event is reachable from the other through the certified dependency edges. After checking acyclicity, the final execution relation is $\prec = \mathrm{TC}(\mathcal{R})$, with optional transitive reduction for storage.

4.3 Event Local Tokenization and Supervision

Given an attributed execution poset G, each event v is expanded into a local token chain

$$B_v = (b_{v,1}, \ldots, b_{v,m_v}). \tag{9}$$

For a supervised decision event $v \in D$, an event bound prediction anchor a_v is placed after its content tokens and followed by the teacher forced target inputs:

$$q_{v,1} = a_v, \qquad q_{v,j} = y_{v,j-1}, \quad j > 1. \tag{10}$$

The label $y_{v,j}$ is attached to prediction row $q_{v,j}$. Event dependencies induce the corresponding token dependencies, producing a token DAG

$$\widetilde{G} = (T, \preceq_{\widetilde{G}}). \tag{11}$$

Standard next token supervision can change when concurrent event blocks exchange positions in the packed sequence.

4.4 Ancestor Attention and Causal Rank Positions

After token expansion, the client may choose any legal packing $\sigma \in \mathrm{LE}(\widetilde{G})$. Let $\sigma(i)$ denote the abstract token stored at tensor row i. The attention mask is defined as

$$M^{\sigma}_{ij} = \begin{cases} 0, & \sigma(j) \preceq_{\widetilde{G}} \sigma(i), \\ -\infty, & \text{otherwise.} \end{cases} \tag{12}$$

Each token can therefore attend only to itself and its causal ancestors. An independent event cannot enter the visible context solely because it appears earlier in the packed sequence.

Ancestor attention removes serialization dependence from the visible context, but physical position IDs can still expose tensor order. We assign each token an intrinsic causal rank

$$\rho(t) = \begin{cases} 0, & \mathrm{Pred}_{\widetilde{G}}(t) = \varnothing, \\ 1 + \max\limits_{s \in \mathrm{Pred}_{\widetilde{G}}(t)} \rho(s), & \text{otherwise.} \end{cases} \tag{13}$$

The client passes $\rho(t)$ through the existing position interface instead of using the tensor row index. Concurrent tokens may share the same rank, while tokens connected by causal dependencies receive increasing ranks. Since $\rho(t)$ depends only on $\widetilde{G}$, the same abstract token receives the same position under every legal packing. All ranks can be computed in $O(|T| + |E_{\widetilde{G}}|)$ time without introducing trainable parameters.

4.5 Federated Update and Local Data

TopoFed-Agent changes only the local representation used during adaptation. For SFT, each client evaluates Eq.3 using masked cross entropy. For policy optimization, the action, log probability, and externally computed advantage are attached to the corresponding decision event, while an execution level verifier reward can be incorporated by the underlying policy optimization method.

After local optimization, client k returns the same update type Δ_k used by the base federated trainer. The server performs

$$\theta_{t+1} = \mathsf{Aggregate}\left(\theta_t, \{\Delta_k : k \in S_t\}\right). \tag{14}$$

The aggregation rule itself is unchanged, so TopoFed-Agent can be used with full parameter adaptation, LoRA, and other parameter efficient updates.

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate TopoFed-Agent in both controlled and interactive settings. For controlled evaluation, we construct four execution structures: fork join, two chains, layered, and unequal chains. We enumerate their legal serializations when feasible and additionally vary the concurrency width from 2 to 7. This setting isolates serialization effects because the execution graph, event contents, targets, and model parameters remain fixed. For end to end agent adaptation, we use AppWorld (Trivedi et al., 2024), which provides executable multi application tasks and state based verification,

Table 1: Sensitivity to legal serialization across four execution structures. Δ_z, $\Delta_{\mathcal{L}}$, and Δ_g denote relative deviations of event-aligned logits, loss, and full parameter gradients, respectively.

	Fork join			Two chains			Layered			Unequal chains			Arbitrary LE
Method	Δ_z	$\Delta_{\mathcal{L}}$	Δ_g	Δ_z	$\Delta_{\mathcal{L}}$	Δ_g	Δ_z	$\Delta_{\mathcal{L}}$	Δ_g	Δ_z	$\Delta_{\mathcal{L}}$	Δ_g	
Sequential	4.82×10^{-2}	6.12×10^{-2}	0.46±0.08	4.35×10^{-2}	5.76×10^{-2}	0.44±0.09	2.61×10^{-2}	3.44×10^{-2}	0.26±0.04	3.97×10^{-2}	5.18×10^{-2}	0.42±0.04	✓
Mask-only	3.12×10^{-2}	4.71×10^{-2}	0.38±0.04	2.95×10^{-2}	4.36×10^{-2}	0.38±0.05	1.72×10^{-2}	2.57×10^{-2}	0.19±0.03	2.88×10^{-2}	4.10×10^{-2}	0.37±0.03	✓
Rank-only	2.17×10^{-2}	3.05×10^{-2}	0.26±0.06	1.84×10^{-2}	2.68×10^{-2}	0.22±0.06	1.31×10^{-2}	1.92×10^{-2}	0.17±0.04	1.56×10^{-2}	2.34×10^{-2}	0.19±0.04	✓
Global-shift	$<10^{-12}$	0.90±0.30	2.71±1.14	$<10^{-12}$	0.74±0.18	2.03±0.47	$<10^{-12}$	0.61±0.17	1.85±0.48	$<10^{-12}$	0.55±0.13	1.57±0.39	✓
Canonical-LE	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	×
TopoFed-Agent	$\mathbf{1.8\times10^{-16}}$	$\mathbf{1.3\times10^{-16}}$	$\mathbf{2.3\times10^{-16}}$	$\mathbf{1.5\times10^{-16}}$	$\mathbf{1.1\times10^{-16}}$	$\mathbf{2.0\times10^{-16}}$	$\mathbf{1.2\times10^{-16}}$	$\mathbf{9.7\times10^{-17}}$	$\mathbf{1.6\times10^{-16}}$	$\mathbf{1.6\times10^{-16}}$	$\mathbf{1.2\times10^{-16}}$	$\mathbf{2.1\times10^{-16}}$	✓

and Gaia2 (Froger et al., 2026), which contains dynamic and asynchronous agent scenarios with action level verifiers. AppWorld is used for federated training and in domain evaluation, while Gaia2 provides an additional asynchronous evaluation setting. Runtime relations are obtained only from program order, tool call and result links, explicit result consumption, and read/write conflicts. We do not infer dependencies from textual similarity or timestamps.

Federated setting. Unless otherwise specified, we use $K = 20$ clients, sample $M = 5$ clients per round, and train for 50 communication rounds. To isolate scheduler induced heterogeneity, all clients share the same execution distribution and differ only in the serialization distribution. At each topological sorting step, client k selects an event v from the current ready set $\mathcal{A}$ according to $P_k(v \mid \mathcal{A}) = \frac{\exp(\beta s_{k,v})}{\sum_{u\in\mathcal{A}} \exp(\beta s_{k,u})}$, where $s_{k,v} \sim \mathcal{N}(0,1)$ is fixed for each client and β controls scheduler preference. We use $\beta \in \{0, 0.5, 1.0, 1.5, 2.0, 2.5\}$. We further vary the client number over $K \in \{8, 16, 32, 64\}$ with a fixed 50% participation rate. For mixed heterogeneity, task instances are partitioned across clients using a Dirichlet distribution with concentration $\alpha = 0.5$.

Baselines. We compare TopoFed-Agent with representative federated, agent adaptation, and serialization baselines. *Seq-FedAvg* applies FedAvg (McMahan et al., 2017) directly to sequentialized agent traces, while *FedProx* (Li et al., 2020) further regularizes local updates under client heterogeneity. FedAgent (Chen et al., 2026a) represents federated agent training under heterogeneous tasks and environments. We further include Fed-GVPO (Dai et al., 2026) and Fed-Agent-Omit (Ning et al., 2026), where their original local agent training objectives are placed in the same federated protocol. For both methods, we retain the released local training procedures and modify only the client partition and federated aggregation wrapper. Canonical-LE converts each execution into a fixed deterministic topological order, whereas Random-LE samples one legal serialization for each local update. We also evaluate order averaging with $B \in \{2, 4, 8, 16, 24\}$ legal serializations to study whether repeated serialization can reduce order-induced learning variation.

Implementation and reproducibility. Controlled experiments use the two layer, two head Transformer described in Section 1, with hidden size 16, feed forward size 32, float64 computation, and dropout disabled. We run five seeds $\{0, 1, 2, 3, 4\}$. For end to end experiments, we use Qwen2.5-1.5B-Instruct as the main backbone(Yang et al., 2024). All gradient based methods use LoRA (Hu et al., 2022) with rank 16, scaling factor 32, and zero dropout. Training uses bfloat16 on four NVIDIA A100 80GB GPUs. End to end results are averaged over seeds $\{42, 43, 44\}$. For recent baselines, we retain their released local objectives and method specific optimizer settings, while matching the backbone, client partition, communication rounds, participation ratio, and environment interaction budget. We save the generated execution posets, scheduler scores, client partitions, random seeds, and complete configuration for every reported run.

5.2 Schedule Invariance under Legal Serializations

We evaluate whether different legal serializations of the same execution lead to different learning signals. Four controlled execution structures are considered, including fork join, two chains, layered, and unequal chains, with 24, 6, 4, and 10 legal serializations, respectively. As shown in Table 1, Sequential exhibits clear serialization sensitivity, with Δ_g ranging from 0.26 to 0.46 across the four structures. Mask-only and Rank-only reduce this variation but remain sensitive because serialization information is still retained in positions or visible contexts. Global-shift further shows that invariant logits alone are insufficient when supervision changes with tensor order. Canonical-LE removes the measured variation by enforcing a fixed topological order, but does not support arbitrary legal serializations. In contrast, TopoFed-Agent preserves arbitrary legal orders while keeping logits, losses, and gradients at numerical precision, confirming that event-local supervision, ancestor attention, and causal-rank positions jointly remove serialization-dependent learning variation.

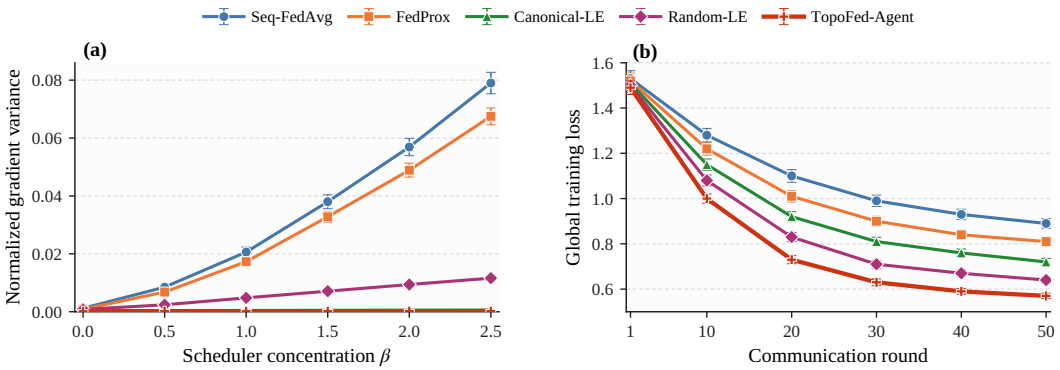


Figure 3: Federated learning under scheduler heterogeneity.

Table 2: End-to-end federated agent performance on AppWorld and Gaia2. TGC and SGC denote Task Goal Completion and Scenario Goal Completion, respectively. $R_{0.75}$ is the number of communication rounds required to reach a validation loss of 0.75.

	AppWorld Test-N		AppWorld Test-C		Gaia2		Federated Optimization	
Method	TGC (%) ↑	SGC (%) ↑	TGC (%) ↑	SGC (%) ↑	Pass@1 (%) ↑	Verifier Score ↑	$R_{0.75}$ ↓	Final Loss ↓
Seq-FedAvg	20.8±0.9	11.6±0.7	9.7±0.6	3.8±0.4	10.9±0.8	0.35±0.01	> 50	0.84±0.02
FedProx	22.4±1.0	12.8±0.8	10.9±0.7	4.5±0.4	11.7±0.7	0.37±0.02	48	0.77±0.02
FedAgent	25.6±0.8	15.2±0.7	13.1±0.8	5.9±0.5	13.6±0.9	0.40±0.02	42	0.72±0.02
Fed-GVPO	29.8±1.1	18.9±0.9	16.7±0.8	8.1±0.6	16.4±0.8	0.45±0.02	34	0.65±0.01
Fed-Agent-Omit	28.7±0.9	18.0±0.8	15.9±0.7	7.6±0.5	15.8±0.7	0.44±0.02	36	0.67±0.02
Canonical-LE	30.5±0.8	19.6±0.7	17.5±0.8	8.7±0.5	17.2±0.8	0.47±0.01	31	0.62±0.01
Random-LE	31.9±0.9	21.0±0.8	18.6±0.9	9.5±0.6	18.3±0.9	0.49±0.02	28	0.60±0.01
TopoFed-Agent	**35.8±0.8**	**24.7±0.7**	**22.1±0.8**	**12.2±0.5**	**21.9±0.7**	**0.54±0.01**	**23**	**0.56±0.01**

5.3 Federated Learning under Scheduler Heterogeneity

We evaluate federated adaptation when clients share the same semantic execution distribution but differ in their scheduler induced serialization distributions. We use $K = 20$ clients with a 25% participation rate and vary $\beta \in \{0, 0.5, 1.0, 1.5, 2.0, 2.5\}$, where larger β induces stronger client specific serialization preferences. As shown in Figure 3(a), the normalized between client gradient variance of Seq-FedAvg and FedProx increases with β, while Random-LE only partially reduces this effect. Canonical-LE remains stable by enforcing a fixed topological order, whereas TopoFed-Agent keeps the variance near numerical zero across all scheduler settings. Figure 3(b) further shows that, under $\beta = 2.5$, TopoFed-Agent reaches a lower training loss more rapidly than the compared methods. These results indicate that scheduler induced serialization differences can affect federated optimization, while serialization invariant local adaptation removes this source of disagreement.

5.4 End-to-End Federated Agent Performance

We evaluate end-to-end federated agent adaptation on AppWorld (Trivedi et al., 2024) and Gaia2 (Froger et al., 2026) under the same client partition, participation rate, communication budget, backbone, and interaction budget. As shown in Table 2, TopoFed-Agent achieves the strongest task performance across both benchmarks. Compared with Random-LE, it improves AppWorld Test-C TGC from 18.6% to 22.1% and Gaia2 Pass@1 from 18.3% to 21.9%. It also reaches the target validation loss in fewer communication rounds and obtains the lowest final training loss. These results show that removing serialization-dependent learning variation improves both federated optimization and downstream agent performance.

5.5 Ablation and Robustness

Figure 4 reports the 2^3 ablation over Ancestor Mask (M), Causal Rank (R), and Event local Target (T) across four execution structures. Removing any component leaves residual serialization sensitivity in the context, position, or supervision, while the full MRT configuration keeps logits, losses, and gradients at numerical precision. This confirms that the three components are jointly required for serialization invariant learning.

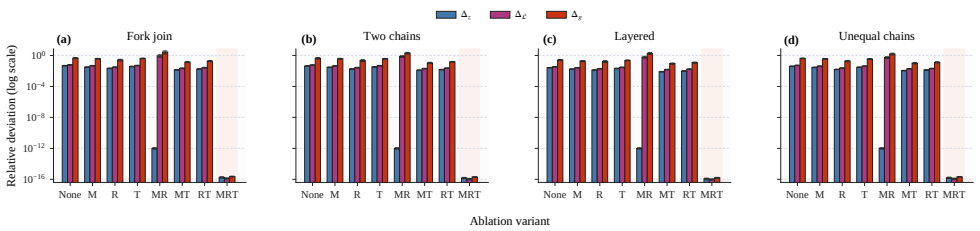


Figure 4: Ablation of the three components in TopoFed-Agent across four execution structures.

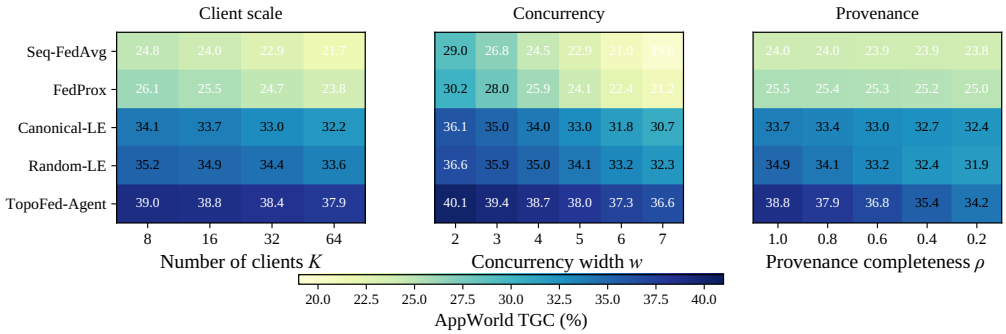


Figure 5: Robustness of federated agent adaptation under changing system conditions.

Table 3: Efficiency and system cost under the same backbone and LoRA configuration.

Method	Local Time (s) ↓	Peak GPU (GB) ↓	Poset Build (ms) ↓	Mask/Rank (ms) ↓	Comm./Round (MB)	Server Mem. (GB)	Trainable Params. (M)
Seq-FedAvg	18.42±0.21	17.26±0.04	0.00±0.00	0.00±0.00	24.80±0.00	3.21±0.02	6.55
FedProx	18.87±0.24	17.31±0.05	0.00±0.00	0.00±0.00	24.80±0.00	3.21±0.02	6.55
Canonical-LE	18.65±0.20	17.29±0.04	1.16±0.05	0.00±0.00	24.80±0.00	3.21±0.02	6.55
Random-LE	18.73±0.23	17.30±0.04	1.18±0.06	0.00±0.00	24.80±0.00	3.21±0.02	6.55
TopoFed-Agent	19.21±0.25	17.68±0.06	1.24±0.05	2.37±0.09	24.80±0.00	3.21±0.02	6.55

Figure 5 further evaluates robustness to client number K, concurrency width w, and provenance completeness ρ. TopoFed-Agent remains stable as the number of clients and concurrent branches increases. When ρ decreases, fewer independence relations can be certified, so the recovered poset becomes more conservative and performance degrades gradually rather than introducing unsupported concurrency. These results show that TopoFed-Agent remains effective under larger federated scales, higher execution concurrency, and incomplete provenance.

5.6 Efficiency and System Cost

Table 3 reports the computational and system overhead under the same backbone and LoRA configuration. Compared with Seq-FedAvg, TopoFed-Agent increases local training time from 18.42 s to 19.21 s and peak GPU memory from 17.26 GB to 17.68 GB, corresponding to 4.3% runtime and 2.4% memory overhead. Poset construction and mask/rank generation require only 1.24 ms and 2.37 ms per execution, while communication per round, server memory, and trainable parameter count remain unchanged. These results show that TopoFed-Agent introduces limited local preprocessing cost without additional communication or server-side overhead.

6 Conclusion

This paper studies interleaving heterogeneity in federated post training of asynchronous LLM agents and proposes TopoFed-Agent to remove serialization dependent learning variation through runtime certified execution posets, ancestor attention, causal rank positions, and event local supervision. The current method relies on reliable provenance information and is evaluated mainly under synchronous federation and representative execution patterns. Future work will consider incomplete provenance, asynchronous federated training, longer agent trajectories, and broader multi agent and multimodal environments.

AI USE STATEMENT

Generative AI tools were used only for language editing and polishing. All technical content, methods, experiments, theoretical analysis, and conclusions were developed and verified by the authors.

REPRODUCIBILITY STATEMENT

We provide the main experimental configuration, benchmark settings, federated protocol, baselines, and implementation details in Section 5.1. Additional details on runtime provenance, execution-poset construction, controlled execution structures, client partitioning, scheduler generation, metric definitions, model configuration, and rerun procedures are provided in Appendix A and Appendix B. Complete proofs of the theoretical results and additional diagnostic experiments are also included in the appendix. We record random seeds, client partitions, scheduler scores, generated execution posets, and resolved run configurations for each experiment to support reproducibility.

ETHICS STATEMENT

This work studies federated adaptation of language-model agents without centralizing client execution traces. The experiments use established benchmarks and controlled execution settings and do not involve human subjects or newly collected personal data. The proposed method is intended to improve the consistency of distributed agent training while keeping execution traces local to clients.

REFERENCES

Adam AlSayyad, Kelvin Yuxiang Huang, and Richik Pal. AgentTrace: A structured logging framework for agent system observability. *arXiv preprint arXiv:2602.10133*, 2026. URL https://arxiv.org/abs/2602.10133.

Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0461. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/1a134b50202088aa8c595cc99b310e5a-Abstract-Conference.html.

Abhijit Chakraborty, Yash Shah, and Vivek Gupta. Synapse: Federated tool routing via typed compendium artifacts. *arXiv preprint arXiv:2602.00911*, 2026. URL https://arxiv.org/abs/2602.00911.

Canyu Chen, Kangyu Zhu, Zhaorun Chen, Zhanhui Zhou, Shizhe Diao, Yiping Lu, Tian Li, Manling Li, and Dawn Song. Is decentralized LLM agent RL robust to heterogeneity? an asymmetric tale. *arXiv preprint*, 2026a. URL https://fed-agent.github.io/.

Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. Cordon: Semantic transactions for tool-using LLM agents. *arXiv preprint arXiv:2606.17573*, 2026b. URL https://arxiv.org/abs/2606.17573.

Silong Dai, Changzhi Sun, Haolun Wu, Huanran Zheng, Tao Ji, Junchi Yan, Yuanbin Wu, Dell Zhang, Xiaoling Wang, and Xuelong Li. Group verification-based policy optimization for interactive coding agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RY47Tq0VsV.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex-Attention: A programming model for generating fused attention variants. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/61a9278dfef5f871b5e472389f8d6fa1-Abstract-Conference.html.

Zehao Dong, Muhan Zhang, Fuhai Li, and Yixin Chen. PACE: A parallelizable computation encoder for directed acyclic graphs. In *Proceedings of the 39th International Conference on Machine*

Learning, volume 162 of *Proceedings of Machine Learning Research*, pp. 5360–5377. PMLR, 2022. URL https://proceedings.mlr.press/v162/dong22b.html.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres 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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9gw03JpKK4.

Bowei He, Yankai Chen, Xiaokun Zhang, and Xue Liu. Branching policy optimization: Sandbox-native language agent reinforcement learning. *arXiv preprint arXiv:2607.14171*, 2026. URL https://arxiv.org/abs/2607.14171.

Yuchao Hou. FedWorld: Scope-aware federation of agent world models. *arXiv preprint arXiv:2608.01561*, 2026. URL https://arxiv.org/abs/2608.01561.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Sander J. J. Leemans, Sebastiaan J. van Zelst, and Xixi Lu. Partial-order-based process mining: A survey and outlook. *Knowledge and Information Systems*, 65:1–29, 2023. doi: 10.1007/s10115-022-01777-3. URL https://doi.org/10.1007/s10115-022-01777-3.

Chentao Li and Han Guo. Hasse diagrams for attention: A partial order framework for designing transformer masks. *arXiv preprint arXiv:2606.09951*, 2026. URL https://arxiv.org/abs/2606.09951.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pp. 429–450, 2020. URL https://proceedings.mlsys.org/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html.

Yu Li, Sizhe Tang, and Tian Lan. Reason in chains, learn in trees: Self-rectification and grafting for multi-turn agent policy optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 4673–4686. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.229. URL https://aclanthology.org/2026.findings-acl.229/.

Ji Liu, Jiaxiang Ren, Ruoming Jin, Zijie Zhang, Yang Zhou, Patrick Valduriez, and Dejing Dou. Fisher information-based efficient curriculum federated learning with large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10497–10523. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.587. URL https://aclanthology.org/2024.emnlp-main.587/.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ooROvpmxMV.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train ANY AI agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025. URL https://arxiv.org/abs/2508.03680.

Isham Kalappurackal Mansoor, Abhishek Phadke, and Pratip Rana. Verified tool calls improve LLM agent reliability under non-atomic failures. *arXiv preprint arXiv:2608.02645*, 2026. URL https://arxiv.org/abs/2608.02645.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Yansong Ning, Jun Fang, Naiqiang Tan, and Hao Liu. Agent-Omit: Adaptive context omission for efficient LLM agents. *arXiv preprint arXiv:2602.04284*, 2026. URL https://arxiv.org/abs/2602.04284.

OpenTelemetry Authors. OpenTelemetry Specification, version 1.60.0. Cloud Native Computing Foundation, 2026. URL https://github.com/open-telemetry/opentelemetry-specification/tree/v1.60.0. Accessed 2026-08-29.

Kunjal Panchal, Nisarg Parikh, Sunav Choudhary, Lijun Zhang, Yuriy Brun, and Hui Guan. Thinking forward: Memory-efficient federated finetuning of language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2207. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/7fc914993440219b64254e0c27964e11-Abstract-Conference.html.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c66e1fcc9691aae706250638f36f681b-Abstract-Conference.html.

Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41473–41497. PMLR, 2024. URL https://proceedings.mlr.press/v235/qin24a.html.

Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LkFG3lB13U5.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1g2JnRcFX.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan O. Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/dd745a0c4f91fe91866fe6788be9cc28-Abstract-Conference.html.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Veronika Thost and Jie Chen. Directed acyclic graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2101.07965.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/7181-attention-is-all-you-need.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. FLoRA: Federated fine-tuning large language models with heterogeneous low-rank adaptations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0708. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28312c9491d60ed0c77f7fff4ad86dd1-Abstract-Conference.html.

World Wide Web Consortium. Trace context level 2. W3C Candidate Recommendation Draft, March 2024. URL https://www.w3.org/TR/trace-context-2/.

Feijie Wu, Zitao Li, Yaliang Li, Bolin Ding, and Jing Gao. FedBiOT: LLM local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3345–3355, 2024. doi: 10.1145/3637528.3671897. URL https://doi.org/10.1145/3637528.3671897.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! LLM agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1556–1572. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.93. URL https://aclanthology.org/2024.emnlp-main.93/.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Chenchen Zhang. Reinforcement learning for LLM-based multi-agent systems through orchestration traces. *arXiv preprint arXiv:2605.02801*, 2026. URL https://arxiv.org/abs/2605.02801.

Jia-Ying Zheng, Hainan Zhang, Lingxiang Wang, Wangjie Qiu, Hong-Wei Zheng, and Zhi-Ming Zheng. Safely learning with private data: A federated learning framework for large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5293–5306. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.303. URL https://aclanthology.org/2024.emnlp-main.303/.

APPENDIX

This appendix provides additional method details, experimental protocols and reproducibility information, theoretical analysis, and further experiments supporting the main paper.

A Additional Method Details

This appendix expands the parts of TopoFed-Agent that are compressed in the main paper. We keep the same notation throughout. Each local execution is represented by an attributed execution poset

$$G = (V, \prec, \{B_v\}_{v \in V}, D, \{y_v\}_{v \in D}, r), \tag{15}$$

and all event contents, dependency records, targets, and provenance remain on the client. The server only receives the update produced by the underlying federated optimizer.

A.1 Runtime records and dependency certificates

Each runtime event is associated with an execution identifier, an event identifier, an event type, its payload, and provenance fields that can certify dependencies. The payload contains the information that was available to the agent at that event, including textual outputs, structured return values, status codes, or other task state. Schedule dependent attributes such as duration or timeout are retained when they can affect the agent decision. They are treated as node-bound attributes rather than as ordering evidence.

The constructor uses native runtime relations rather than semantic similarity. The relations used by the method are listed in Table 4. Message and tracing identifiers play the same role as causal context in distributed tracing systems (OpenTelemetry Authors, 2026; World Wide Web Consortium, 2024). Textual similarity and timestamps are never sufficient by themselves to certify a dependency or an independence relation.

A.2 Conservative execution-poset construction

Let $\tau = (v_1, \ldots, v_{|V|})$ be the observed trace, $\mathcal{E}^{\mathrm{dep}}$ the certified dependency edges, and $\mathcal{I}^{\mathrm{ind}}$ the pairs whose independence is positively certified. The client begins with $\mathcal{E}^{\mathrm{dep}}$. For a pair that is still unrelated, the observed order is removed only when that pair has an independence certificate. Otherwise, the order in τ is retained. This gives

$$\mathcal{R} = \mathcal{E}^{\mathrm{dep}} \cup \left\{ (u, v) : u \lhd_\tau v,\ u \parallel_{\mathcal{E}^{\mathrm{dep}}} v,\ (u, v) \notin \mathcal{I}^{\mathrm{ind}} \right\}, \tag{16}$$

followed by an acyclicity check and transitive closure. Algorithm 2 gives the complete procedure.

The construction deliberately prefers extra order to unsupported concurrency. When provenance is incomplete, unresolved pairs remain ordered as observed. This can reduce the number of legal serializations but does not create a parallel relation that the runtime has not certified. If the supplied dependency records themselves are inconsistent and create a cycle, the execution is excluded from the invariance test rather than repaired by a heuristic rule.

A.3 Token-DAG expansion and event-local supervision

For event v, let

$$B_v = (b_{v,1}, \ldots, b_{v,m_v}) \tag{17}$$

be its content tokens. Tokens inside one event form a local chain. If $v \in D$ is supervised, an event-bound prediction anchor a_v is placed after the event content and the teacher-forced prediction rows are

$$q_{v,1} = a_v, \qquad q_{v,j} = y_{v,j-1}, \quad j > 1. \tag{18}$$

Row $q_{v,j}$ receives target $y_{v,j}$. The supervision key is the tuple $(\texttt{execution-id}, \texttt{event-id}, j)$ rather than a global token offset. Therefore, swapping incomparable event blocks does not move a target to another abstract prediction row.

Algorithm 2 Conservative runtime-certified poset construction

Require: Observed trace τ, dependency edges $\mathcal{E}^{\mathrm{dep}}$, independent pairs $\mathcal{I}^{\mathrm{ind}}$
$\mathcal{R} \leftarrow \mathcal{E}^{\mathrm{dep}}$
Compute reachability under $\mathcal{R}$
for each pair (u, v) with $u \lhd_\tau v$ **do**
  if u and v are already comparable under $\mathcal{R}$ **then**
    continue
  else if $(u, v) \in \mathcal{I}^{\mathrm{ind}}$ or $(v, u) \in \mathcal{I}^{\mathrm{ind}}$ **then**
    leave u and v unordered
  else
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(u, v)\}$
  end if
end for
Verify that $\mathcal{R}$ is acyclic
$\prec \leftarrow \mathrm{TC}(\mathcal{R})$
Optionally apply transitive reduction for storage
return $G = (V, \prec, \{B_v\}, D, \{y_v\}, r)$

Table 4: Runtime evidence used by the conservative poset constructor.

Evidence	Relation	Role in the execution poset
Program order	earlier local step $\rightarrow$ later local step	Preserves mandatory order inside one sequential control path.
Message send/receive	send $\rightarrow$ receive	Connects causally related events across workers or tools.
Tool call/result	call $\rightarrow$ result	Prevents a result from preceding the invocation that created it.
Explicit consumption	result $\rightarrow$ decision	Records that a later decision actually consumed a specific result.
Fork/join metadata	fork $\rightarrow$ branch and branch $\rightarrow$ join	Identifies branch creation and synchronization.
Versioned read/write	write $\rightarrow$ dependent read/write	Preserves conflicts when a later event observes a particular state version.
Certified independence	unordered pair	Removes an observed order only when the runtime can establish independence.

For every event relation $u \prec v$, the last token of u precedes the first token of v. If v has no content token, the edge terminates at a_v. Combining these cross-event edges with the local token chains gives the token DAG $\widetilde{G} = (T, \preceq_{\widetilde{G}})$. Teacher-forced target inputs stay inside the same decision-event chain and do not introduce edges from unrelated events.

A.4 Packing, padding, and ancestor-closed truncation

A legal packing is any $\sigma \in \mathrm{LE}(\widetilde{G})$. Packing changes only the physical row assigned to an abstract token. Padding is added after packing and is excluded from both the loss and the attention context. When several executions share one batch, the attention matrix is block diagonal across executions so that a token cannot attend to another execution.

Physical tail truncation is not serialization invariant because different legal packings can place different abstract events near the end of the tensor. We therefore select a subgraph before packing. The selected token set must be ancestor closed: if a supervised token is retained, every causal ancestor needed by that token is also retained. Decision events are selected first together with their ancestor closures. If capacity remains, additional events are added by a node-bound rule that does not depend on the chosen legal packing. The same abstract subgraph is then packed under every serialization.

Table 5: Controlled structures used for the serialization-invariance experiments.

Structure	Construction	Legal serializations
Fork join	One start event, four mutually incomparable branch events, and one join event depending on all four branches.	$4! = 24$
Two chains	Two independent chains of length two. Legal orders are all interleavings that preserve the order inside each chain.	$\binom{4}{2} = 6$
Layered	Two incomparable events in the first layer, two incomparable events in the second layer, and every first-layer event preceding every second-layer event.	$2! \times 2! = 4$
Unequal chains	Two independent chains of lengths two and three.	$\binom{5}{2} = 10$

A.5 Ancestor attention and causal-rank positions

For a legal packing σ, the attention mask is

$$M^{\sigma}_{ij} = \begin{cases} 0, & \sigma(j) \preceq_{\widetilde{G}} \sigma(i), \\ -\infty, & \text{otherwise.} \end{cases} \tag{19}$$

Thus, each token sees itself and its causal ancestors, but not an incomparable event that happens to occupy an earlier tensor row.

The causal rank is computed directly on the token DAG,

$$\rho(t) = \begin{cases} 0, & \mathrm{Pred}_{\widetilde{G}}(t) = \varnothing, \\ 1 + \max_{s \in \mathrm{Pred}_{\widetilde{G}}(t)} \rho(s), & \text{otherwise.} \end{cases} \tag{20}$$

Since $\rho(t)$ is a property of the abstract token, the same token receives the same position index under every legal packing. For models with rotary position embeddings, including the Qwen2.5 family used in the end-to-end experiments (Yang et al., 2024; Su et al., 2024), the RoPE rotation is indexed by $\rho(t)$ instead of the physical tensor row. Section C.4 shows that this preserves the required permutation equivariance.

A.6 Local objectives and federated interface

For SFT, the client evaluates the event-local masked cross-entropy loss defined in the main paper. A policy or verifier-weighted objective uses the same prediction rows, but attaches its weight to the corresponding decision event before the local gradient is computed. The event target, action metadata, log probability, and verifier signal therefore move together as node-bound attributes.

The method does not change the type of model update returned to the server. Full-parameter training and LoRA (Hu et al., 2022) can use the same local representation, and the resulting update can be consumed by FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020), or an adaptive server optimizer such as FedAdam (Reddi et al., 2021). No execution graph, event identifier, dependency edge, or local trace corpus is uploaded.

B Experimental Protocol and Reproducibility

B.1 Controlled execution structures

The four controlled structures are chosen so that legal serializations can be enumerated exactly while covering distinct concurrency patterns. Across serializations, the execution graph, event contents, prediction targets, and parameter snapshot are fixed. Table 5 gives the exact combinatorial structure.

The controlled Transformer has two layers, two attention heads, hidden size 16, feed-forward size 32, float64 arithmetic, and dropout disabled. We use seeds $\{0, 1, 2, 3, 4\}$. The model is not trained in the

invariance test. For each execution, every legal serialization is materialized from the same parameter snapshot, one forward and backward pass is performed, and outputs are aligned by abstract token identity before logits, losses, and full-parameter gradients are compared.

B.2 Interactive benchmarks and evaluation

AppWorld. AppWorld provides executable multi-application tasks with programmatic state-based verification (Trivedi et al., 2024). We use the same Test-N and Test-C evaluation protocol reported in the main paper. Task Goal Completion and Scenario Goal Completion are computed with the benchmark evaluator. Task instances are assigned to clients before scheduler perturbations are introduced.

Gaia2. Gaia2 contains dynamic and asynchronous agent scenarios with action-level verification (Froger et al., 2026). We use it as an additional asynchronous evaluation environment and report Pass@1 together with the mean verifier score. The compared methods share the same backbone, tool interface, interaction budget, and client partition.

Runtime relation extraction. Program order and tool-call/result links are always retained. Explicit consumption is recorded only when the runtime exposes a direct use relation, and read/write conflicts are constructed from versioned environment state. We do not infer causal edges from textual similarity or raw timestamp order.

B.3 Federated partitions and scheduler distributions

The default setting uses $K = 20$ clients, $M = 5$ sampled clients per round, and 50 communication rounds. The default participation rate is therefore 25%. End-to-end results use seeds $\{42, 43, 44\}$. For the client-scale experiment, $K \in \{8, 16, 32, 64\}$ and the participation rate is fixed at 50%.

For scheduler-only experiments, all clients share the same semantic execution distribution. Client k receives a fixed score $s_{k,v} \sim \mathcal{N}(0, 1)$ for each abstract event. At each topological-sorting step, an event is sampled from the current ready set $\mathcal{A}$ according to

$$P_k(v \mid \mathcal{A}) = \frac{\exp(\beta s_{k,v})}{\sum_{u \in \mathcal{A}} \exp(\beta s_{k,u})}. \tag{21}$$

The score table is fixed throughout a run, so increasing β strengthens a persistent client-specific preference rather than introducing a fresh random order every round. We use $\beta \in \{0, 0.5, 1.0, 1.5, 2.0, 2.5\}$.

For mixed semantic and scheduler heterogeneity, task families are partitioned with a Dirichlet distribution. If $\boldsymbol{\pi}_c$ denotes the client proportions for task family c, then

$$\boldsymbol{\pi}_c \sim \mathrm{Dirichlet}(\alpha \mathbf{1}_K). \tag{22}$$

The main experimental setting uses $\alpha = 0.5$. The additional joint study in Section D.3 varies $\alpha \in \{0.1, 0.5, 1.0\}$ and $\beta \in \{0, 1.0, 2.5\}$.

B.4 Baselines and parity conditions

Seq-FedAvg applies FedAvg directly to sequentialized traces. FedProx uses the same trace representation with its proximal local objective. FedAgent, Fed-GVPO, and Fed-Agent-Omit retain their released local objectives while being placed under the same client partition, participation schedule, communication-round budget, backbone, and environment-interaction budget. Canonical-LE maps each execution to one deterministic topological order. Random-LE samples one legal topological order for each local update. Order averaging evaluates $B \in \{2, 4, 8, 16, 24\}$ legal serializations from the same parameter snapshot and averages the resulting gradients before the optimizer step.

For recent baselines, method-specific optimizer choices that are intrinsic to the released objective are retained. Shared variables such as the backbone, LoRA parameterization, client partition, number of rounds, participation ratio, and interaction budget are held fixed. This avoids replacing a method-specific training objective with a common optimizer that was not part of the original baseline.

Table 6: Run-level information stored for reproducibility.

Field	Stored value
Model	Backbone checkpoint identifier, tokenizer identifier, LoRA rank, scale, and dropout.
Federation	K, selected clients per round, communication rounds, client-selection sequence, aggregation rule.
Data	Benchmark split, task-to-client partition, partition seed, and partition hash.
Scheduler	β, fixed client-event score table, generated legal packing identifiers.
Poset construction	Dependency records, independence certificates, transitive-reduction flag, provenance-completeness setting.
Optimization	Local objective, optimizer class, learning-rate schedule, batch and accumulation settings, clipping, and method-specific coefficients.
Precision/hardware	Arithmetic type, GPU model, GPU count, deterministic-kernel flags when enabled.
Evaluation	Decoding/tool budget, verifier configuration, metric implementation, and evaluation seed.
Artifacts	Generated posets, per-round metrics, model-update hashes, final task outputs, and verifier outputs.

B.5 Model configuration and run manifest

The end-to-end backbone is Qwen2.5-1.5B-Instruct (Yang et al., 2024). All gradient-based methods use LoRA with rank 16, scaling factor 32, and zero LoRA dropout. End-to-end training uses bfloat16 on four NVIDIA A100 80GB GPUs. Controlled invariance tests use float64 on the two-layer Transformer described above.

To make a run exactly recoverable, every experiment stores a resolved configuration rather than relying on implicit package defaults. Table 6 lists the fields that are saved for each run. Optimizer-specific values that differ between recent baselines are recorded in the manifest together with the source configuration from the corresponding released implementation.

B.6 Metric definitions

For a fixed execution with legal serialization set Σ, let x^σ denote an event-aligned quantity and let $\bar{x} = |\Sigma|^{-1} \sum_{\sigma \in \Sigma} x^\sigma$. We compute

$$\Delta_z = \frac{1}{|\Sigma|} \sum_{\sigma \in \Sigma} \frac{\|z^\sigma - \bar{z}\|_2}{\|\bar{z}\|_2 + 10^{-12}}, \tag{23}$$

$$\Delta_{\mathcal{L}} = \frac{1}{|\Sigma|} \sum_{\sigma \in \Sigma} \frac{|\mathcal{L}^\sigma - \bar{\mathcal{L}}|}{|\bar{\mathcal{L}}| + 10^{-12}}, \tag{24}$$

$$\Delta_g = \frac{1}{|\Sigma|} \sum_{\sigma \in \Sigma} \frac{\|g^\sigma - \bar{g}\|_2}{\|\bar{g}\|_2 + 10^{-12}}. \tag{25}$$

The full gradient is flattened in a fixed parameter-name order. For LoRA experiments, the same definition is applied to the trainable LoRA tensors only.

At round r, the normalized between-client gradient variance is

$$V_r = \frac{\sum_{k \in S_r} \|g_k^r - \bar{g}^r\|_2^2}{\sum_{k \in S_r} \|g_k^r\|_2^2 + 10^{-12}}, \qquad \bar{g}^r = \frac{1}{|S_r|} \sum_{k \in S_r} g_k^r. \tag{26}$$

Efficiency measurements include the complete local forward, backward, and optimizer step and exclude data loading. Poset construction and mask/rank construction are timed separately. Commu-

nication is measured from the serialized update payload rather than inferred only from parameter counts.

B.7 Rerun procedure

A rerun reconstructs the saved client partition and scheduler score table, reloads the same global initialization, and replays the recorded client-selection sequence. For each selected client, the poset is reconstructed from the stored provenance record, the legal packing is generated from the saved packing identifier, and the local objective is evaluated with the saved optimizer configuration. Controlled invariance tests additionally store the abstract token identifiers used to align logits and gradients across legal serializations. This separates differences caused by model computation from differences caused by data ordering or client sampling.

C Theoretical Analysis

C.1 Soundness of conservative reconstruction

Let $\prec^\star$ denote the true causal order of an execution and let τ be an observed trace that is a linear extension of $\prec^\star$.

Proposition 1 (Conservative reconstruction)**.** *Assume every edge in $\mathcal{E}^{\mathrm{dep}}$ is a true dependency and every pair in $\mathcal{I}^{\mathrm{ind}}$ is truly incomparable under $\prec^\star$. Then the recovered order $\prec$ satisfies*

$$\prec^\star \subseteq \prec . \tag{27}$$

The reconstruction may add order when provenance is incomplete, but it does not remove a true dependency.

Proof. Take any pair $u \prec^\star v$. Because τ is a legal trace of the true execution, $u \lhd_\tau v$. If u already reaches v through $\mathcal{E}^{\mathrm{dep}}$, the relation is present before the conservative-completion step. Otherwise, the pair is unresolved. Since u and v are truly comparable, sound independence certification implies that neither (u, v) nor (v, u) belongs to $\mathcal{I}^{\mathrm{ind}}$. Equation 16 therefore retains the observed relation (u, v). Transitive closure cannot remove this relation. Hence every true causal pair remains comparable in the recovered poset. □

The proposition explains why missing evidence and incorrect evidence behave differently. Missing independence evidence makes the recovered poset more ordered. A false independence certificate can remove a true causal relation and violates the assumption of Proposition 1.

C.2 Permutation representation of legal packings

Fix a token DAG $\widetilde{G} = (T, \preceq)$ with $n = |T|$. A legal packing σ is a bijection from physical rows $\{1, \ldots, n\}$ to abstract tokens. For two legal packings σ and σ', let $P \in \{0, 1\}^{n \times n}$ be the permutation matrix that maps the row order of σ to that of σ'.

All token attributes used by TopoFed-Agent are node-bound. Token embeddings, event-type features, target identifiers, and causal ranks therefore move with the abstract token. If $H^{0,\sigma}$ is the input hidden-state matrix under packing σ, then

$$H^{0,\sigma'} = PH^{0,\sigma}. \tag{28}$$

Lemma 1 (Mask conjugacy)**.** *The ancestor masks of two legal packings satisfy*

$$M^{\sigma'} = PM^{\sigma}P^{\top}. \tag{29}$$

Proof. Let row i' in σ' contain the same abstract token as row i in σ, and define j' analogously. By construction,

$$M^{\sigma'}_{i'j'} = 0 \iff \sigma'(j') \preceq \sigma'(i') \tag{30}$$

$$\iff \sigma(j) \preceq \sigma(i) \tag{31}$$

$$\iff M^{\sigma}_{ij} = 0. \tag{32}$$

The same correspondence holds for masked entries, giving the stated conjugacy relation. □

Lemma 2 (Causal-rank invariance)**.** *For every abstract token t, the value $\rho(t)$ is identical under all legal packings of $\widetilde{G}$.*

Proof. The recurrence defining $\rho(t)$ depends only on the predecessor set $\mathrm{Pred}_{\widetilde{G}}(t)$. A legal packing changes row order but not the predecessor relation. Starting from source tokens and proceeding in any topological order therefore gives the same value for every token. □

C.3 Permutation equivariance of masked self-attention

Consider one Transformer attention head (Vaswani et al., 2017). For hidden states H,

$$Q = HW_Q, \qquad K = HW_K, \qquad V = HW_V, \tag{33}$$

and the masked score matrix is

$$A = \frac{QK^\top}{\sqrt{d}} + M. \tag{34}$$

Under a legal row permutation, $H' = PH$ and Lemma 1 gives $M' = PMP^\top$. Because the projection matrices are shared across rows,

$$Q' = PQ, \qquad K' = PK, \qquad V' = PV. \tag{35}$$

Hence

$$A' = \frac{PQK^\top P^\top}{\sqrt{d}} + PMP^\top \tag{36}$$

$$= PAP^\top. \tag{37}$$

Row-wise softmax commutes with simultaneous row and column permutation,

$$\mathrm{softmax}(PAP^\top) = P\,\mathrm{softmax}(A)P^\top. \tag{38}$$

Therefore,

$$\mathrm{Attn}(H', M') = \mathrm{softmax}(A')V' \tag{39}$$

$$= P\,\mathrm{softmax}(A)P^\top PV \tag{40}$$

$$= P\,\mathrm{Attn}(H, M). \tag{41}$$

Multi-head concatenation and the output projection are row-shared, so the relation holds for the complete attention sublayer. Residual addition, layer normalization, MLPs, and the output head are also row-shared and therefore preserve the same permutation relation.

C.4 Extension to rotary position embeddings

Qwen2.5 uses rotary position embeddings. Let R_m denote the rotation associated with position index m (Su et al., 2024; Yang et al., 2024). Under TopoFed-Agent, abstract token t uses $m = \rho(t)$. By Lemma 2, the same token has the same rank under every legal packing. Let $\mathcal{R}_\rho$ denote the row-wise RoPE operator under packing σ. Under σ', the operator is the row-permuted version

$$\mathcal{R}'_\rho = P\mathcal{R}_\rho P^\top. \tag{42}$$

For the query matrix,

$$Q'_{\mathrm{rope}} = \mathcal{R}'_\rho(PQ) \tag{43}$$

$$= P\mathcal{R}_\rho P^\top PQ \tag{44}$$

$$= PQ_{\mathrm{rope}}, \tag{45}$$

and the same argument gives $K'_{\mathrm{rope}} = PK_{\mathrm{rope}}$. The score matrix therefore still transforms as $A' = PAP^\top$, so the attention-equivariance argument is unchanged. Physical row indices would not satisfy this property because the rotation assigned to an incomparable event would change when its tensor row changes.

C.5 LINEAR-EXTENSION INVARIANCE

Theorem 1 (Schedule-invariant prediction, loss, and gradient)**.** *Fix G, its event-local supervision, and token DAG $\widetilde{G}$. Assume node-bound token attributes, ancestor attention, causal-rank positions, row-shared Transformer sublayers, deterministic computation, and an ancestor-closed selected subgraph. For any two legal packings $\sigma, \sigma' \in \mathrm{LE}(\widetilde{G})$ and every supervised decision token t,*

$$z^{\sigma}_{\theta,t} = z^{\sigma'}_{\theta,t}, \tag{46}$$

$$\mathcal{L}^{\sigma}_{\mathrm{poset}}(\theta; G) = \mathcal{L}^{\sigma'}_{\mathrm{poset}}(\theta; G), \tag{47}$$

and

$$\nabla_\theta \mathcal{L}^{\sigma}_{\mathrm{poset}}(\theta; G) = \nabla_\theta \mathcal{L}^{\sigma'}_{\mathrm{poset}}(\theta; G). \tag{48}$$

Proof. Equation 28 establishes the permutation relation at the input. Lemma 1 gives the corresponding mask permutation. Sections C.3 and C.4 show that one attention sublayer maps row-permuted inputs to row-permuted outputs. Residual addition and layer normalization preserve this relation, and the MLP applies the same parameters to every row. Thus, if $H^{\ell,\sigma'} = PH^{\ell,\sigma}$ before layer $\ell + 1$, then the complete layer gives

$$H^{\ell+1,\sigma'} = PH^{\ell+1,\sigma}. \tag{49}$$

By induction, the relation holds at every layer. After undoing the row permutation, the hidden state of each abstract decision token is identical. A row-shared output head therefore produces identical logits for that token.

Event-local supervision attaches the same target and the same loss weight to the same abstract prediction row. Both packings therefore sum exactly the same token-level loss terms, so the scalar loss functions are identical for every value of θ. Two identical scalar functions have identical derivatives wherever differentiable. Standard Transformer components are differentiable almost everywhere, which yields the stated gradient equality. □

The equalities are exact over real arithmetic. Parallel floating-point reductions can accumulate values in different orders, so GPU implementations may exhibit differences at numerical precision even when the abstract computation is invariant. This is why the controlled float64 experiment and the large-model bfloat16 validation use different numerical tolerances.

C.6 STOCHASTIC OPERATIONS AND TRUNCATION BOUNDARIES

Ordinary dropout is sampled by physical tensor position. Two legal packings can therefore receive different realized dropout masks even though their mask distributions are identical. Exact sample-wise invariance can be restored by deriving each random key from node-bound identifiers such as $(\texttt{execution-id}, \texttt{event-id}, \texttt{token-offset}, \texttt{layer-id})$. The invariance experiments in the main paper instead disable dropout, which directly satisfies the deterministic-computation assumption.

A similar issue arises from physical truncation. Keeping the first L tensor rows can select different abstract events under two legal packings. The ancestor-closed selection rule in Section A.4 removes this dependence by selecting the abstract subgraph before packing.

C.7 FEDERATED GRADIENT HETEROGENEITY

Let

$$g_k^{\mathrm{seq}} = \mathbb{E}_{G\sim\mu_k,\, \sigma\sim q_k(\cdot|G)} \left[\nabla \mathcal{L}_{\mathrm{seq}}(\theta; G, \sigma)\right] \tag{50}$$

be the expected sequential gradient of client k. Introduce a common reference scheduler $r(\sigma \mid G)$ only for analysis and define

$$h_k = \mathbb{E}_{G\sim\mu_k,\, \sigma\sim r(\cdot|G)} \left[\nabla \mathcal{L}_{\mathrm{seq}}(\theta; G, \sigma)\right], \tag{51}$$

$$\delta_k = g_k^{\mathrm{seq}} - h_k. \tag{52}$$

The term h_k captures the client difference that would remain under a common serialization distribution, while δ_k captures the shift induced by the actual scheduler. Let

$$\bar{h} = \sum_k p_k h_k, \qquad \bar{\delta} = \sum_k p_k \delta_k. \tag{53}$$

Then

$$\zeta^2_{\text{seq}} = \sum_k p_k \left\| g_k^{\text{seq}} - \sum_j p_j g_j^{\text{seq}} \right\|_2^2 \tag{54}$$

$$= \sum_k p_k \left\| (h_k - \bar{h}) + (\delta_k - \bar{\delta}) \right\|_2^2 \tag{55}$$

$$= \sum_k p_k \|h_k - \bar{h}\|_2^2 + \sum_k p_k \|\delta_k - \bar{\delta}\|_2^2 \tag{56}$$

$$+ 2 \sum_k p_k \left\langle h_k - \bar{h}, \delta_k - \bar{\delta} \right\rangle. \tag{57}$$

The interaction term can have either sign. We therefore do not claim that removing scheduler dependence must reduce the numerical variance of every arbitrary sequential objective.

For TopoFed-Agent, Theorem 1 removes the dependence on $q_k(\sigma \mid G)$ at the execution level.

Corollary 1 (Scheduler-only expected-gradient heterogeneity vanishes)**.** *If all clients share the same semantic execution distribution $\mu_k = \mu$ and differ only in their serialization distributions, then every client's expected TopoFed-Agent gradient is identical. Consequently,*

$$\zeta^2_{\text{poset}} = 0. \tag{58}$$

Proof. For a fixed execution G, Theorem 1 gives a gradient $g(\theta; G)$ that is the same for every legal serialization. Taking expectation over any client-specific $q_k(\sigma \mid G)$ therefore leaves the same $g(\theta; G)$. Since the semantic distribution μ is also shared, all clients have the same expected gradient after averaging over G. □

When semantic data distributions differ, the remaining heterogeneity need not vanish. The method removes only the component caused by legal serialization and leaves genuine task or data differences to the underlying federated optimizer.

C.8 Implication for local SGD

We state the consequence in the same nonconvex local-SGD form used in standard analyses (Stich, 2019). Let F_k be the client objective induced by the poset representation and $F = \sum_k p_k F_k$. Assume each F_k is L-smooth, stochastic gradients are unbiased with conditional variance at most σ^2, M clients are sampled per round, and each selected client performs H local steps of size η with $\eta L H$ sufficiently small.

At communication round r, smoothness gives

$$F(\theta^{r+1}) \le F(\theta^r) + \left\langle \nabla F(\theta^r), \theta^{r+1} - \theta^r \right\rangle + \frac{L}{2} \|\theta^{r+1} - \theta^r\|_2^2. \tag{59}$$

The aggregated update can be written as an average of the H local stochastic gradients. Add and subtract $\nabla F_k(\theta^r)$ inside each local gradient. This separates the update error into stochastic noise, client-objective mismatch, and local drift away from θ^r. Unbiasedness controls the first term by σ^2/M. Define the remaining semantic heterogeneity by

$$\zeta^2_{\text{rem}} \ge \sum_k p_k \|\nabla F_k(\theta) - \nabla F(\theta)\|_2^2 \tag{60}$$

for the iterates under consideration. L-smoothness bounds the local-drift term by a quantity proportional to $\eta^2 L^2 H^2 (\sigma^2 + \zeta^2_{\text{rem}})$. Substituting these bounds into Eq. 59, absorbing the cross terms with

Young's inequality, and summing over R rounds gives universal constants $C_i > 0$ such that

$$\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\|\nabla F(\theta^r)\|_2^2 \le \frac{C_1(F(\theta^0)-F_{\inf})}{\eta HR} \tag{61}$$

$$+\frac{C_2 L\eta\sigma^2}{M} + C_3 L^2\eta^2 H^2(\sigma^2+\zeta_{\mathrm{rem}}^2). \tag{62}$$

The form is the usual local-SGD tradeoff between optimization progress, stochastic noise, and residual client drift. Under scheduler-only heterogeneity, Corollary 1 gives $\zeta_{\mathrm{rem}} = 0$. No term depending on the serialization distributions q_k remains. Under mixed data and scheduler heterogeneity, the semantic component remains and the bound does not claim otherwise.

C.9 Why deterministic canonicalization is not equivalent

A deterministic canonicalizer c maps each poset to one chosen linear extension $c(G)$. If every client applies the same c, then repeated observations of the same G use the same sequence and within-rule scheduler variation disappears. This does not make the sequential objective a function of the poset alone. Two valid canonicalizers c and c' may order incomparable events differently. In ordinary sequential training, those two sequences can produce different preceding contexts and physical positions, so in general

$$\mathcal{L}_{\mathrm{seq}}(\theta; c(G)) \neq \mathcal{L}_{\mathrm{seq}}(\theta; c'(G)). \tag{63}$$

Canonicalization therefore replaces a runtime-dependent arbitrary order with a rule-dependent arbitrary order. By contrast, Theorem 1 implies that TopoFed-Agent gives the same event-aligned loss for every $c(G) \in \mathrm{LE}(G)$.

C.10 Why averaging several legal orders is not equivalent

Let $g(\sigma)$ be the sequential gradient for a legal serialization sampled from a fixed distribution and let $\mu_g = \mathbb{E}[g(\sigma)]$. For B independent legal serializations,

$$\bar{g}_B = \frac{1}{B}\sum_{b=1}^{B} g(\sigma_b). \tag{64}$$

Then

$$\mathbb{E}\|\bar{g}_B - \mu_g\|_2^2 = \frac{1}{B^2}\mathbb{E}\left\|\sum_{b=1}^{B}(g(\sigma_b)-\mu_g)\right\|_2^2 \tag{65}$$

$$= \frac{1}{B^2}\sum_{b=1}^{B}\mathbb{E}\|g(\sigma_b)-\mu_g\|_2^2 \tag{66}$$

$$= \frac{1}{B}\mathbb{E}\|g(\sigma)-\mu_g\|_2^2, \tag{67}$$

where the cross terms vanish by independence and zero mean. Order averaging therefore reduces serialization variance at the familiar $1/B$ rate, but requires B forward/backward evaluations. TopoFed-Agent instead targets invariance using a single legal packing.

C.11 Computational and communication cost

Causal ranks are computed by a topological dynamic program in $O(|T| + |E_{\widetilde{G}}|)$ time. After a reachability index is available, checking all unresolved event pairs is $O(|V|^2)$; straightforward transitive closure or reduction gives an $O(|V|^3)$ worst case for the complete conservative constructor. In practice, event graphs are much smaller than token graphs and the preprocessing is performed once per local execution.

A dense implementation materializes an $|T| \times |T|$ mask, which has the same asymptotic storage shape as ordinary dense causal attention. Sparse or block-mask backends can skip disallowed query-key blocks (Dong et al., 2025). Communication volume, server-side model state, trainable-parameter count, number of local optimizer steps, and aggregation payload format are unchanged because the method adds no trainable server component.

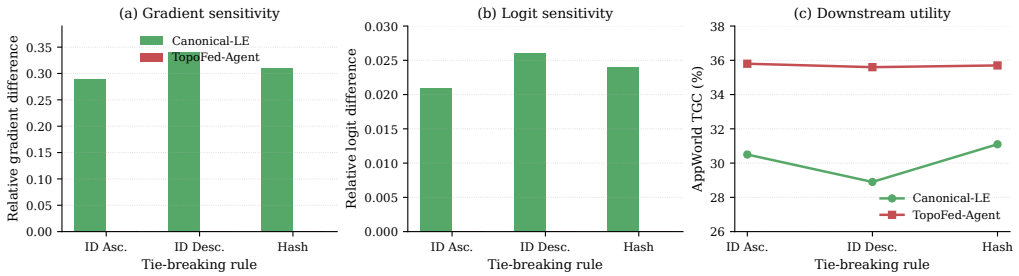


Figure 6: Canonical-LE sensitivity to three deterministic tie-breaking rules. The experiment separates invariance to the observed runtime order from invariance to the arbitrary canonical rule itself.

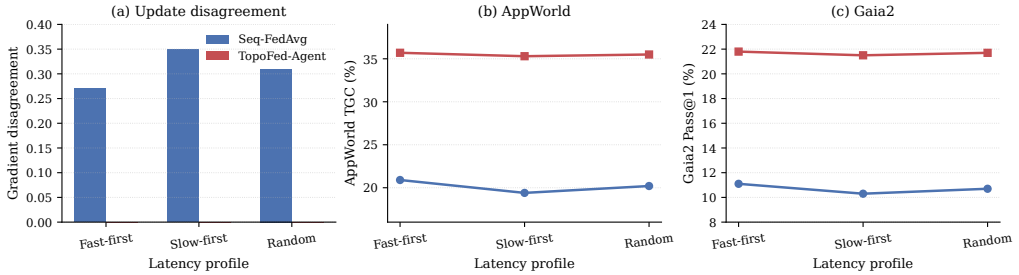


Figure 7: Trace replay with different latency profiles while preserving tool outputs, dependencies, and terminal state.

D Additional Experiments and Diagnostics

D.1 Canonical-LE sensitivity to tie-breaking

Canonical-LE is deterministic only after a tie-breaking rule is chosen. We test three rules on the same execution posets: ascending event ID, descending event ID, and a hash order from the execution and event identifiers. The execution graph and supervision remain fixed. For each rule, Canonical-LE and TopoFed-Agent are evaluated from the same parameter snapshot. We report event-aligned logit difference, full-gradient difference, and AppWorld TGC.

As shown in Figure 6, this diagnostic tests whether changing only the canonical rule changes the learned sequential objective. A rule-dependent shift for Canonical-LE would explain why a method can have near-zero within-rule Δ_g in the main table yet still differ in downstream training. TopoFed-Agent should be insensitive to the rule because the rule controls physical packing but not target, ancestor context, or causal-rank position.

D.2 Trace replay under asynchronous tool latency

The scheduler experiment in the main paper samples legal orders directly. We additionally perturb completion order through tool latency while holding tool outputs and terminal state fixed. Each selected execution is first cached. During replay, delays are inserted only for certified independent tool calls. We use three profiles: Fast-first, Slow-first, and Random-latency. A dependent consumer still waits for the result on which it depends. We compare Seq-FedAvg and TopoFed-Agent using gradient disagreement, AppWorld TGC, and Gaia2 Pass@1.

Figure 7 tests whether the phenomenon persists when the order difference is generated by an asynchronous runtime rather than by direct topological sampling. If only the completion order changes, any additional gradient disagreement measures sensitivity to scheduling rather than a change in task semantics.

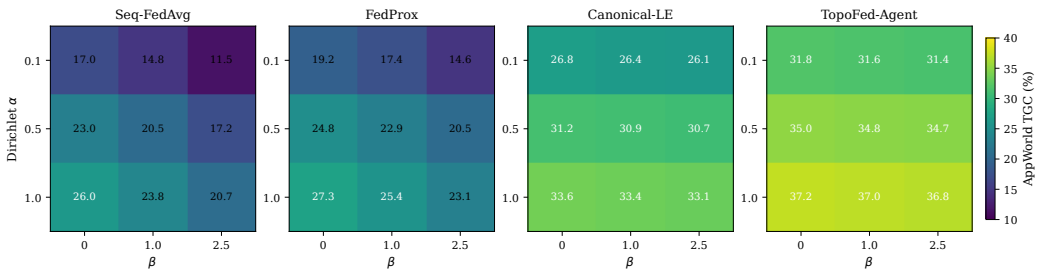


Figure 8: AppWorld TGC under joint semantic and scheduler heterogeneity. Lower α indicates stronger task skew and larger β indicates stronger scheduler preference.

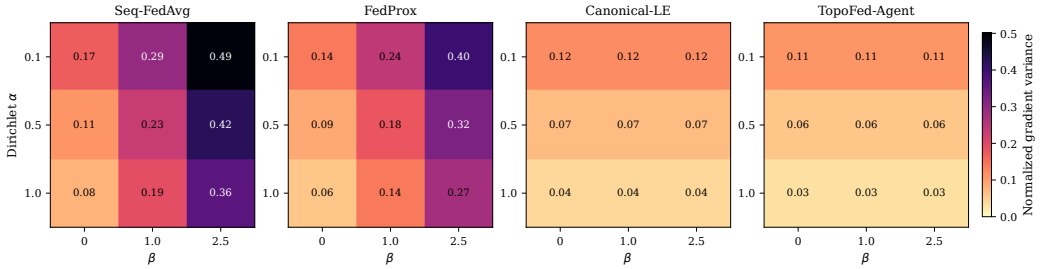


Figure 9: Normalized between-client gradient variance under joint semantic and scheduler heterogeneity.

D.3 Joint scheduler and data heterogeneity

We combine task-family skew and scheduler preference using $\alpha \in \{0.1, 0.5, 1.0\}$ and $\beta \in \{0, 1.0, 2.5\}$. All combinations use the same number of communication rounds and the same client-sampling rule. Seq-FedAvg, FedProx, Canonical-LE, and TopoFed-Agent are compared using AppWorld TGC and normalized between-client gradient variance.

Figures 8 and 9 summarize the joint effect of semantic and scheduler heterogeneity. TopoFed-Agent is not expected to remove genuine semantic heterogeneity. The relevant diagnostic is whether increasing β adds substantial disagreement on top of the variation already caused by α. This directly matches the decomposition in Eq. 57.

D.4 Provenance errors beyond incompleteness

The main paper varies provenance completeness. We additionally separate two error modes. In the missing-dependency condition, a fraction of dependency records is hidden from the constructor and no false independence certificate is added. The conservative rule therefore retains observed order for unresolved pairs. In the false-independence condition, a fraction of truly comparable pairs is incorrectly inserted into $\mathcal{I}^{\mathrm{ind}}$. Corruption rates are $\{0, 5, 10, 20\}\%$. We report task score, gradient error relative to the clean-provenance update, and the causal violation rate, defined as the fraction of supervised decision contexts that omit at least one true ancestor.

Figure 10 compares the two provenance error modes. Missing information should mainly make the recovered poset more conservative. False independence can remove a true causal edge, so it can change the visible context itself and is the more serious violation of the method assumptions.

D.5 Large-model invariance with Qwen2.5 and LoRA

The exact machine-precision test in the main paper uses a small float64 Transformer. To verify the implementation on the end-to-end architecture, we select executions with multiple legal serializations, freeze the Qwen2.5-1.5B base model, and compute gradients only for the rank-16 LoRA parameters. Dropout is disabled. For each execution, several legal packings are evaluated from the same parameter

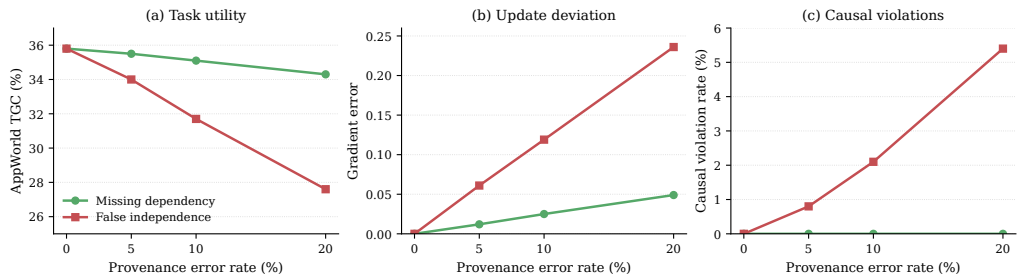


Figure 10: Diagnostics under missing dependency information and false independence certificates.

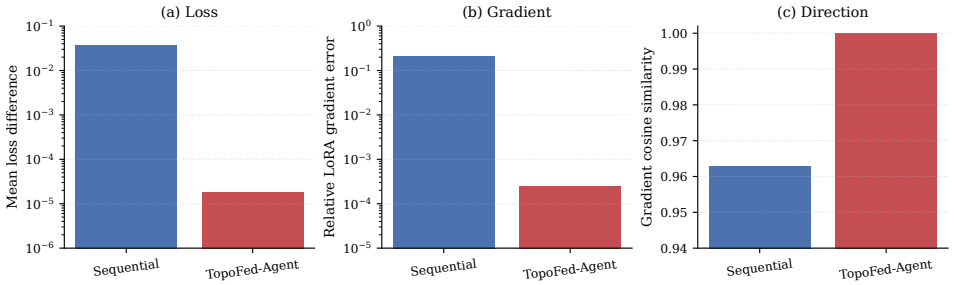


Figure 11: Large-model serialization diagnostics on Qwen2.5-1.5B-Instruct with LoRA. Finite-precision GPU kernels are evaluated with practical numerical tolerances rather than the float64 machine-precision threshold.

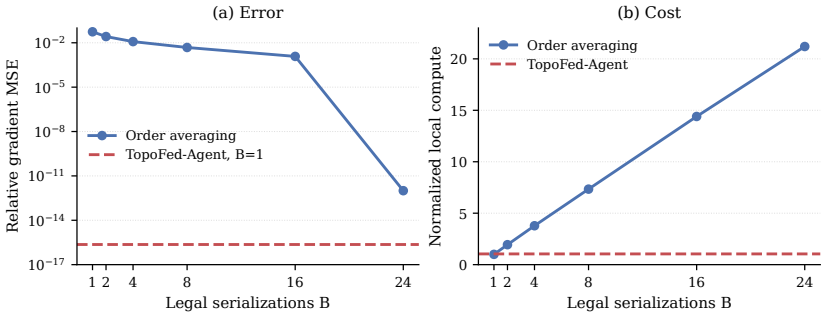


Figure 12: Error and local-compute tradeoff for order averaging.

snapshot. We report the mean loss difference, relative LoRA-gradient error, and pairwise gradient cosine similarity.

Figure 11 reports the large-model invariance diagnostics. This test covers RoPE, bfloat16 arithmetic, and LoRA-only gradients, which are absent from the controlled two-layer model. The theory predicts invariance of the abstract computation when causal ranks are used as the RoPE indices; small residual numerical differences can still arise from finite-precision reductions.

D.6 Cost-benefit of order averaging

Order averaging is a natural alternative because averaging more legal serializations reduces the scheduler-induced component of a sequential gradient. We evaluate $B \in \{1, 2, 4, 8, 16, 24\}$, report gradient error relative to the all-order reference, and record normalized local compute. TopoFed-Agent uses one legal packing per local update.

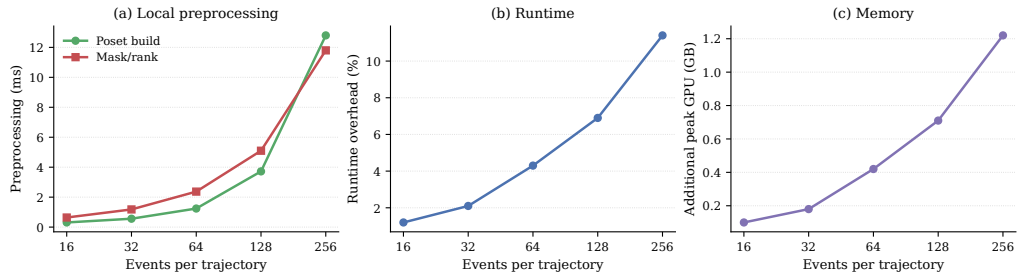


Figure 13: Local preprocessing and system overhead as the number of events per trajectory increases.

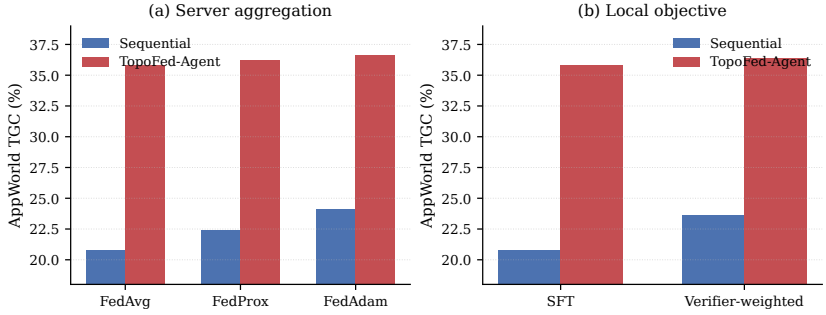


Figure 14: Compatibility of the execution-poset representation with different aggregation rules and local objectives.

Figure 12 complements the $1/B$ variance calculation in Section C.10. It separates a statistical reduction obtained by repeated forward/backward evaluations from a representation that is designed to make one legal packing sufficient.

D.7 Trajectory scale

The main robustness experiment varies concurrency width. We also vary the total number of events per trajectory while keeping the concurrent-event ratio fixed. Event counts are $\{16, 32, 64, 128, 256\}$. We record poset-construction time, mask/rank time, total runtime overhead, and additional peak GPU memory.

Figure 13 examines how local preprocessing and system overhead change with trajectory length. The most relevant boundary is not the combinatorial number of linear extensions, because TopoFed-Agent does not enumerate them, but the size and density of the recovered execution graph and token DAG.

D.8 Compatibility with aggregation and local objectives

Finally, we test whether the representation can be combined with different server aggregation rules and local learning objectives. The aggregation study compares FedAvg, FedProx-style aggregation, and FedAdam (Reddi et al., 2021). The local-objective study compares SFT with an event-bound verifier-weighted surrogate. The same execution-poset construction, ancestor attention, causal ranks, and event-local supervision are used in every case.

Figure 14 evaluates compatibility with different aggregation rules and local objectives. The purpose is not to identify a universally best optimizer, but to check the interface claim made in the main paper: the representation change is local to the client, while the server continues to consume an ordinary model update.