ActivityEditor: Learning to Synthesize Physically Valid Daily Activity Schedules

Anonymous ACL submission

Abstract

Synthesizing realistic 24-hour activity schedules from socio-demographic profiles is essential for urban applications yet remains challenging in data-scarce regions where no historical travel records exist. While large language models (LLMs) have shown promise for zero-shot activity generation, single-pass architectures lack self-correction mechanisms and frequently violate physical constraints such as temporal coverage, continuity, and duration validity. We propose **ActivityEditor**, a dual-LLM-agent framework that reframes activity-chain synthesis as an iterative Generate-then-Edit task. An intention-driven agent maps demographic profiles to a coarse activity skeleton, which is then iteratively refined by an editor agent through targeted operations (Add, Delete, Shift, Replace) that enforce physical and logical constraints. The editor agent is aligned via Group Relative Policy Optimization (GRPO) with rule-based rewards, enabling it to internalize schedule-level behavioral constraints through exploration rather than imitation. Extensive cross-state experiments on the National Household Travel Survey demonstrate that ActivityEditor achieves strong cross-state zero-shot transfer within the NHTS framework, significantly improving temporal consistency and constraint satisfaction over both traditional generators and LLM-based methods. Our code is available at: `https://anonymous.4open.science/r/ActivityEditor-066B`.

1 Introduction

Generating realistic daily activity schedules from socio-demographic profiles is a fundamental task in travel demand modeling (Liao et al., 2025). Given a person's attributes—age, employment status, household composition, and other census-level descriptors—the goal is to synthesize a complete 24-hour activity chain (an ordered sequence of activities with types and start/end times) that is both statistically faithful and physically valid. High-quality synthetic schedules underpin a broad range of urban applications, from transportation planning (Zheng et al., 2023) to epidemic response (Gao et al., 2017), yet obtaining real activity data remains difficult due to privacy regulations (De Montjoye et al., 2013), high survey costs, and severe regional imbalance in data availability.

In practice, many target regions—newly planned districts, emerging cities, or areas subject to strict data regulations—have no historical activity records available for model training. This motivates *zero-shot cross-regional synthesis*: learning transferable activity regularities from a data-rich source region and applying them to unseen targets without any local calibration data. However, existing approaches to activity-chain synthesis, whether rule-based activity models (Bowman and Ben-Akiva, 2001; Goulias et al., 2011) or learning-based generators (Liao et al., 2025), most require region-specific calibration data and cannot generalize to unseen populations without retraining.

Recent work has turned to large language models (LLMs) for activity schedule generation, exploiting their embedded world knowledge about human behavior. Early attempts leverage LLMs' role-playing ability (Park et al., 2023): by prompting the model to assume the persona of a city resident with a given demographic profile, it directly generates plausible daily routines (Wang et al., 2024a). To improve behavioral coherence, subsequent methods introduce structured reasoning, CoPB (Shao et al., 2024), for example, decomposes generation through behavioral-theory-guided prompts to elicit more logically consistent schedules. Moving beyond pure prompting, more recent work applies post-training to align LLM outputs with domain-specific mobility patterns: TravelReasoner (Liu et al., 2026) constructs reasoning-aligned training data from real travel records and

employs curriculum-based supervised fine-tuning with reward-filtered self-reinforcement to improve generation fidelity.

Despite encouraging progress, these methods share two fundamental limitations. First, they all adopt single-pass generation that lacks mechanisms to detect and correct constraint violations after output. Errors accumulated over a long 24-hour horizon—temporal overlaps, gaps, or unrealistic durations—remain uncorrected. Second, even post-training approaches are fundamentally imitation-based: they learn what good schedules look like by filtering or imitating high-quality examples, but do not explicitly optimize for physical feasibility as an objective. Consequently, their constraint satisfaction depends entirely on whether training examples cover similar patterns, and degrades noticeably on out-of-distribution profiles or unseen regions.

We propose ACTIVITYEDITOR, a dual-LLM-agent framework that reframes activity-chain synthesis as an iterative **Generate-then-Edit** task. An *Intention-Driven Agent* (frozen LLM) infers latent intentions from user profiles and produces a coarse activity skeleton. An *Editor Agent* then iteratively audits this draft against physical and logical constraints, applying targeted operations (Add, Delete, Shift, Replace) to resolve violations. The Editor Agent is trained through supervised fine-tuning on teacher-generated editing traces, followed by Group Relative Policy Optimization (GRPO) with rule-based rewards that explicitly encode temporal coverage, continuity, and duration validity.

Our contributions are as follows:

- We formalize *zero-shot profile-conditioned activity-chain synthesis* and show that socio-demographic regularities provide strong transferable signals for generating high-fidelity 24-hour schedules for unseen populations, bypassing the need for local historical data.
- We propose ActivityEditor, a dual-agent Generate-then-Edit framework. By separating intention planning from constraint enforcement and aligning the Editor Agent via GRPO, we establish a self-correcting mechanism that substantially improves semantic coherence and schedule-level physical validity.
- Extensive cross-state experiments on the National Household Travel Survey demonstrate that ActivityEditor consistently outperforms both traditional generators and LLM-based methods, achieving strong cross-state zero-shot transferability with significant improvements in temporal consistency and constraint satisfaction.

2 Related Work

2.1 Mobility Modeling

Spatial Mobility Modeling. Prior work on spatial mobility models location-level transitions using RNNs, GANs (Feng et al., 2018, 2020), and LLM-based methods for next-location prediction (Wang et al., 2023; Feng et al., 2025b) or trajectory generation (Wang et al., 2024b; Li et al., 2024). These approaches operate on POI sequences or coordinates and require geographic context. In contrast, our work focuses on non-spatial, profile-conditioned activity-chain synthesis, learning transferable semantic and temporal patterns rather than city-specific spatial behaviors.

Activity-Chain Synthesis. More closely related to our work is research on generating daily activity schedules, ordered sequences of activity types with start/end times, from socio-demographic profiles. Traditional activity-based models (Bowman and Ben-Akiva, 2001; Goulias et al., 2011) and deep generative approaches (Liao et al., 2025) have demonstrated strong in-domain performance but require region-specific calibration data. Recent LLM-based methods bypass this limitation: CoPB (Shao et al., 2024) and RAGHome (Liu et al., 2024) leverage prompting and retrieval-augmented generation to synthesize schedules in zero-shot or few-shot settings, while TravelReasoner (Liu et al., 2026) further applies post-training with reasoning-aligned data to improve generation fidelity. LLMob (Wang et al., 2024a) frames the LLM as an urban resident agent with self-consistency checking.

Despite this progress, existing activity-chain methods share two key limitations. First, they universally adopt single-pass generation without explicit self-correction mechanisms, causing constraint violations (temporal overlaps, gaps, unrealistic durations) to persist in the final output. Second, even fine-tuned models rely on imitation-based objectives that do not directly optimize for physical feasibility. ActivityEditor addresses both limitations through a dual-agent Generate-then-Edit architecture with GRPO-based constraint alignment.

2.2 LLM Alignment and Agentic Reinforcement Learning

Post-training alignment of LLMs has progressed from Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) through Direct Preference Optimization (DPO) (Rafailov et al., 2023) to group-based methods. Group Relative Policy Optimization (GRPO) (Guo et al., 2025) and its variants (Yu et al., 2025) eliminate the need for explicit value functions by normalizing rewards within sample groups, achieving strong performance on complex reasoning tasks.

A parallel trend extends RL beyond single-turn alignment toward *agentic* settings, where LLMs act as autonomous decision-makers over multi-step horizons. Such approaches optimize sequential policies through environment feedback (Zhang et al., 2025; Qi et al., 2024) and emphasize iterative self-correction via tool-use reasoning (Feng et al., 2025a; Dong et al., 2025). Recent work further explores turn-level reward design for multi-turn agent optimization (Zeng et al., 2025; Wei et al., 2025).

ActivityEditor connects these two threads: we apply GRPO—originally designed for single-turn reasoning alignment—within a multi-turn agentic editing loop. The Editor Agent iteratively critiques and revises activity schedules, with rule-based rewards providing step-level feedback on physical constraint satisfaction. This combination enables the model to internalize domain-specific regularities through exploration, going beyond what imitation-based training can achieve.

3 Problem Formulation

We define the task of zero-shot profile-conditioned activity-chain synthesis.

Input. Each individual i is characterized by a socio-demographic profile $D_i = \{d_1, d_2, \ldots, d_K\}$, comprising K attributes such as age, gender, employment status, household income, and vehicle ownership.

Output. A daily activity chain $A_i = \{A_1, A_2, \ldots, A_N\}$, where each activity $A_n = [T_n, S_n, E_n]$ specifies an activity type $T_n \in \mathcal{V}_T$ (e.g., home, work, shopping), a start time S_n, and an end time E_n. The chain must cover a complete 24-hour period with $S_1 = 00{:}00$, $E_N = 24{:}00$, and $E_n = S_{n+1}$ for all consecutive pairs.

Task. Given profiles from an unseen target region $\mathcal{R}_{\text{target}}$, learn a mapping $\mathcal{M} : D_i \rightarrow A_i$ using only training data from a source region $\mathcal{R}_{\text{source}}$. The model must generalize without any target-region adaptation, producing activity chains that are both statistically faithful to real population-level patterns and physically valid (no temporal gaps, overlaps, or implausible durations).

4 Methods

4.1 Overview

As illustrated in Figure 1, ACTIVITYEDITOR comprises two collaborative modules. The *Intention-Driven Agent* (a frozen LLM) maps demographic profiles into a coarse activity draft S_{draft} by reasoning over socio-semantic priors. The *Editor Agent* then iteratively refines S_{draft} through four editing operations—Add, Delete, Shift, and Replace—to enforce physical and logical constraints. The Intention-Driven Agent operates without fine-tuning to preserve universal behavioral priors, while the Editor Agent is optimized through a two-stage pipeline (SFT followed by GRPO) to achieve both socio-semantic coherence and strict physical validity. Prompt templates for both agents are provided in Appendix A.3.4.

4.2 Intention-Driven Agent

The Intention-Driven Agent leverages a frozen LLM to infer latent behavioral intentions from user profiles and produce a preliminary activity skeleton. The key assumption is that the mapping from socio-demographic attributes to daily intentions (e.g., a full-time worker needs commuting; a retiree does not) is consistent across regions, enabling zero-shot transfer.

Input. The agent receives a structured profile D_i with K attributes (e.g., `age_range`, `employment_status`, `relationship`, `primary_activity`).

Skeleton Generation. Given D_i, the agent initiates a reasoning trace within a thought space to infer the user's latent intentions. By leveraging its parametric knowledge of behavioral patterns, the agent articulates the rationale behind the user's daily demands. For example, if `primary_activity = "Retired"`, the agent reasons that no work commute is needed and allocates more time to home and leisure activities. Guided by these inferred intentions, the agent constructs a draft schedule S_{draft}—a chronologically ordered activity chain with approximate time allocations.

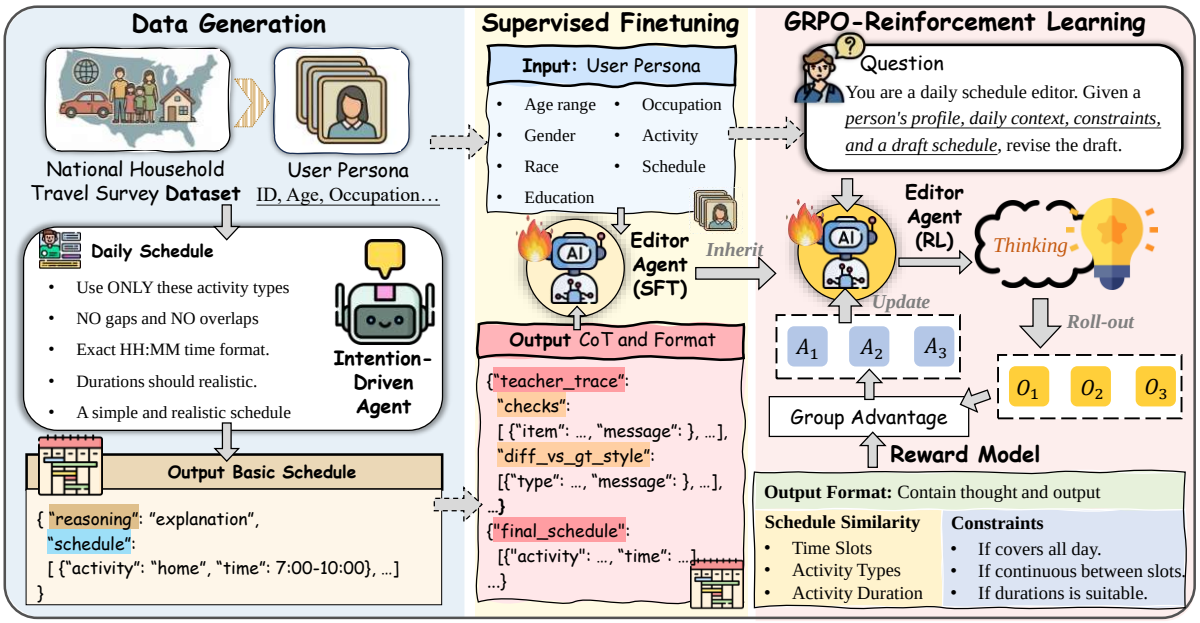

Figure 1: Overview of the ACTIVITYEDITOR framework.

The primary objective of this stage is semantic coherence: S_{draft} captures the macroscopic behavioral pattern consistent with the user's profile, but may contain temporal imprecisions (e.g., slight overlaps or unrealistic durations) that serve as the raw material for the Editor Agent to refine.

4.3 Editor Agent

The Editor Agent bridges the gap between abstract intentions and physical reality. While the Intention-Driven Agent ensures semantic plausibility, the resulting S_{draft} may violate temporal constraints that LLMs do not natively enforce—such as time overlaps, gaps, or implausible durations. The Editor Agent functions as a constraint-aware auditor that iteratively detects violations and applies targeted corrections.

Constraint Checking. The agent first acts as a critic within the [THOUGHT] space, comprehensively auditing the draft against two categories of constraints:

Hard Constraints ensure structural validity:

- *Physical:* The activity chain must seamlessly cover the full 24-hour period (S_1 = 00:00, E_N = 24:00) without gaps or overlaps ($E_n = S_{n+1}$ for all n).
- *Logical:* Consecutive activities must be distinct types; identical adjacent activities are structurally invalid and must be merged into a single continuous segment.

Soft Constraints evaluate behavioral realism:

- *Commonsense:* Activities should align with the user's profile. For example, a user profiled as *retired* should not be assigned regular, prolonged *work* activities.
- *Temporal:* Each activity duration must fall within a physiologically plausible range. While sleeping or working can naturally span several hours, a *drop-off/pick-up* event should not exceed 30–40 minutes.
- *Coherence:* The sequence should exhibit smooth transitions without excessive fragmentation (e.g., alternating home↔shopping every 10 minutes is unrealistic).

Iterative Refinement. Upon detecting violations, the agent triggers a ReAct-style loop (Yao et al., 2022), selecting operations from a predefined action space to locally modify the draft:

- ADD: insert a missing activity to fill temporal gaps;
- DELETE: remove an irrational or profile-inconsistent event;
- SHIFT: adjust start/end times to resolve overlaps or correct durations;
- REPLACE: swap an inappropriate activity type with a contextually suitable one.

After each operation, the agent re-evaluates the updated schedule against all constraints. This critique-act cycle repeats until all hard constraints are satisfied and soft constraints are sufficiently addressed. The agent then outputs the finalized activity chain

in JSON format.

4.4 Training Strategy

The Editor Agent is trained through a two-stage pipeline: supervised fine-tuning (SFT) to establish the editing paradigm, followed by GRPO to internalize physical constraints through exploration.

Data Generation. Real travel surveys record only final executed schedules, lacking the intermediate cognitive processes of constraint checking and conflict resolution. To construct training data for the editing mechanism, we employ a teacher model (GPT-5.2) to reverse-engineer editing traces. Specifically, we supply the teacher with: (1) the user demographic profile D_i, (2) the draft S_{draft} generated by the Intention-Driven Agent, and (3) the corresponding ground-truth schedule. The teacher infers a detailed Chain-of-Thought trace that explicitly identifies which constraints are violated in the draft and articulates the step-by-step editing operations needed to align it with the ground truth. This yields high-quality triplets of (input, reasoning trace, corrected output) for supervised training.

Supervised Fine-Tuning. We fine-tune a student model on the synthesized editing traces to initialize the Editor Agent. This stage addresses the cold-start problem inherent in complex structured generation: by training on paired thought traces and JSON outputs, the model acquires both foundational domain knowledge about human mobility patterns and the structural paradigm of self-critique—learning to open a `[THOUGHT]` space, sequentially propose editing actions, and output a formatted `[JSON]` schedule.

GRPO Alignment. To move beyond imitation and enable the model to internalize physical laws through trial-and-error, we apply GRPO in the second stage. For each training instance, the full dual-agent pipeline is executed: the Intention-Driven Agent produces S_{draft} from a user profile, and the Editor Agent (policy model) generates G candidate refinements in parallel. A programmatic, rule-based reward model evaluates each candidate without human intervention along three dimensions:

Formatting Reward (R_{fmt}): This ensures adherence to the "Generate-then-Edit" paradigm. It assigns scores based on the presence of the reasoning process (`[THOUGHT]`) and the structured output (`[JSON]`):

$$R_{\text{fmt}} = \begin{cases} 1.0, & \text{if both [THOUGHT] and [JSON] exist} \\ 0.5, & \text{if only one tag exists} \\ 0.0, & \text{otherwise} \end{cases} \tag{1}$$

Constraint Reward (R_{con}): This validates the physical validity of the generated schedule. Let $\mathcal{S} = \{(s_1, e_1), (s_2, e_2), \ldots, (s_N, e_N)\}$ denote the generated schedule containing N segments, where s_i and e_i are the start and end times in minutes. R_{con} is the average of three Boolean checks:

$$R_{\text{con}} = \frac{1}{3}\big(\mathbb{I}_{\text{full}} + \mathbb{I}_{\text{cont}} + \mathbb{I}_{\text{dur}}\big) \tag{2}$$

where $\mathbb{I}_{\text{full}}$ ensures full-day coverage; $\mathbb{I}_{\text{cont}}$ enforces temporal continuity without gaps or overlaps; and $\mathbb{I}_{\text{dur}}$ constrains each activity duration to a realistic physiological range.

Fidelity Reward (R_{sim}): This evaluates the statistical realism of the generated sequence compared to standard mobility patterns. It combines the token-level accuracy (Acc), Macro-F1 score, and the Jensen-Shannon Divergence (JSD) of activity type distributions (p_{act}) and interval length distributions (p_{int}):

$$\begin{aligned} R_{\text{sim}} = {} & 0.40 \cdot \text{Acc} + 0.10 \cdot \text{F1}_{\text{macro}} \\ & + 0.25 \cdot \big(1 - \text{JSD}(p^{\text{act}}_{\text{gen}} \parallel p^{\text{act}}_{\text{gt}})\big) \\ & + 0.25 \cdot \big(1 - \text{JSD}(p^{\text{int}}_{\text{gen}} \parallel p^{\text{int}}_{\text{gt}})\big) \end{aligned} \tag{3}$$

where p_{gen} and p_{gt} denote the probability distributions of the generated schedule and the ground truth, respectively.

Based on the composite reward $R = R_{\text{fmt}} + R_{\text{con}} + R_{\text{sim}}$, the group advantage is computed to update the policy. By comparing candidates within each group, the model learns to increase the probability of generating high-quality editing traces that satisfy physical constraints, internalizing domain regularities through its own exploration rather than teacher imitation.

5 Experiments

5.1 Experimental Setup

Dataset. We conduct experiments on the 2017 National Household Travel Survey (NHTS), a nationwide survey containing 24-hour activity and travel diaries from diverse states, households, and socio-demographic groups. Although our experiments use one unified data source, NHTS is not a

Method	California			Arizona			Georgia			Oklahoma		
	Acc ↑	*Mint* ↓	*Atype* ↓	*Acc* ↑	*Mint* ↓	*Atype* ↓	*Acc* ↑	*Mint* ↓	*Atype* ↓	*Acc* ↑	*Mint* ↓	*Atype* ↓
Traditional sequence baselines												
◇ DeepMove	0.554	0.660	0.229	0.521	0.666	0.252	0.448	0.657	0.264	0.472	0.676	0.281
◇ LSTPM	0.402	0.705	0.290	0.420	0.691	0.300	0.379	0.693	0.288	0.376	0.697	0.287
General LLM baselines												
◇ ChatGPT	0.660	0.425	0.199	0.664	0.399	0.195	0.583	0.403	0.201	0.616	0.404	0.158
◇ Qwen	0.553	0.535	0.228	0.551	0.598	0.194	0.511	0.583	0.238	0.542	0.570	0.167
◇ LLaMA	0.526	0.576	0.291	0.524	0.643	0.248	0.486	0.624	0.305	0.515	0.611	0.213
Activity-time baselines												
◇ Bhandari24	0.586	0.514	0.246	0.571	0.531	0.258	0.548	0.552	0.273	0.559	0.539	0.264
◇ TravelReasoner	0.692	0.368	0.171	0.673	0.382	0.136	0.639	0.305	0.209	0.651	0.293	0.138
◇ LLMob	0.643	0.479	0.217	0.636	0.487	0.224	0.612	0.501	0.235	0.625	0.488	0.221
Prompt-based and retrieval-augmented baselines												
◇ CoPB	0.703	0.577	0.141	0.674	0.579	0.157	0.602	0.603	0.152	0.627	0.567	0.140
◇ RAGHome	0.679	0.556	0.183	0.658	0.548	0.198	0.593	0.574	0.167	0.614	0.528	0.142
ActivityEditor variants												
◇ SFT_LLaMA	0.724	0.214	0.110	0.689	0.256	0.158	0.632	0.253	0.195	0.650	0.227	0.165
◇ SFT_Qwen	0.753	0.231	0.104	0.754	**0.202**	0.107	0.698	**0.226**	0.139	0.696	**0.192**	0.138
◇ GRPO_LLaMA	0.732	0.248	0.122	0.720	0.260	0.131	0.681	0.276	0.147	0.690	0.268	0.142
◇ **GRPO_Qwen**	**0.761**	**0.202**	**0.094**	**0.768**	0.215	**0.101**	**0.736**	0.229	**0.119**	**0.743**	0.223	**0.116**

Table 1: Performance comparison across four states under different models and settings.

single-city or single-region dataset, and thus supports controlled cross-state evaluation under a consistent survey framework.

We use California (CA) as the source state for training, validation, and in-domain testing, and hold out Arizona (AZ), Georgia (GA), and Oklahoma (OK) as unseen zero-shot target states. To prevent data leakage, target-state schedules are never used during training, retrieval, prompting, teacher-trace construction, GRPO reward design, or hyperparameter tuning. California is split into train/validation/test sets before teacher-trace generation, and teacher traces are generated only from the California training split. Full dataset statistics, split details, and the leakage-prevention protocol are provided in Appendix A.1.

Baselines and implementation. We compare ActivityEditor with four groups of baselines: (1) traditional sequence models, including DeepMove (Feng et al., 2018) and LSTPM (Sun et al., 2020); (2) prompt-based and retrieval-augmented methods, including CoPB (Shao et al., 2024) and RAGHome (Liu et al., 2024); (3) recent activity-time LLM baselines, including Bhandari24 (Bhandari et al., 2024), TravelReasoner (Liu et al., 2026), and LLMob (Wang et al., 2024a); and (4) general zero-shot LLMs, including ChatGPT-5.2, Qwen3-8B, and LLaMA3.1-8B under single-pass and dual-agent settings. For ActivityEditor, we use GPT-5.2 to construct teacher traces, initialize the Editor Agent by SFT on Qwen3-8B with LoRA, and further optimize it with GRPO using rule-based rewards. Detailed baseline descriptions and hyperparameters are provided in Appendix A.2 and Appendix A.4.

Evaluation metrics. We evaluate generated schedules from four aspects: sequence fidelity, temporal alignment, distributional consistency, and schedule validity. The main reported metrics are slot-level accuracy (Acc), macro-interval divergence (Mint), and activity-type divergence (Atype). We also evaluate additional metrics such as macro-F1, edit distance, BLEU, hour-level divergence, full-day coverage, no-gap/no-overlap validity, duration validity, start/end-home validity, and overall schedule validity. Detailed metric definitions are provided in Appendix A.6.

5.2 Main Results

Table 1 presents the comprehensive performance comparison of all models across the four states.

Model	CA → CA			CA → AZ			CA → GA			CA → OK			AZ → CA		
	Acc ↑	*Mint* ↓	*Atype* ↓	*Acc* ↑	*Mint* ↓	*Atype* ↓	*Acc* ↑	*Mint* ↓	*Atype* ↓	*Acc* ↑	*Mint* ↓	*Atype* ↓	*Acc* ↑	*Mint* ↓	*Atype* ↓
SFT_CoPB	0.743	0.289	0.154	0.682	0.420	0.127	0.642	0.303	0.153	0.711	0.325	**0.114**	0.654	0.371	0.109
SFT_RAGHome	0.691	0.349	0.176	0.671	0.322	0.183	0.643	0.324	0.134	0.672	0.246	0.120	0.636	0.287	0.151
SFT_LLaMA	0.724	0.214	0.110	0.689	0.256	0.158	0.632	0.253	0.195	0.650	0.227	0.165	0.653	0.228	0.141
SFT_Qwen	0.753	0.231	0.104	0.745	0.239	0.113	0.708	0.248	0.130	0.714	0.241	0.127	0.699	0.214	0.097
GRPO_LLaMA	0.732	0.248	0.122	0.720	0.260	0.131	0.681	0.276	0.147	0.690	0.268	0.142	0.671	0.233	0.118
GRPO_Qwen	**0.761**	**0.202**	**0.094**	**0.768**	**0.215**	**0.101**	**0.736**	**0.229**	**0.119**	**0.743**	**0.223**	0.116	**0.721**	**0.191**	**0.086**

Table 2: Cross-state transfer performance.

The results show clear differences in how different model families handle non-spatial activity-chain synthesis.

Traditional sequence baselines, including DeepMove and LSTPM, show limited performance under the zero-shot setting. Although they can model sequential regularities to some extent, they rely heavily on historical activity sequences and therefore struggle to infer a complete daily schedule from static socio-demographic profiles alone. This limitation is especially clear in their high temporal divergence across all four states.

General LLM baselines, including ChatGPT, Qwen, and LLaMA, generate more plausible daily routines than traditional sequence models, but their performance remains limited in temporal alignment and distributional consistency. This suggests that general commonsense knowledge alone is insufficient for reliable activity-chain synthesis. In particular, direct single-pass generation does not explicitly optimize activity duration, ordering, or population-level activity distributions.

Activity-time baselines and prompt-based methods provide stronger task-specific priors. Among the reproducible activity-time baselines, LLMob achieves the strongest overall accuracy, indicating that profile-aware activity generation is helpful for this task. CoPB and RAGHome further improve activity-type divergence in several states, showing that prompt structure and retrieval can recover coarse semantic activity regularities. However, these methods still exhibit relatively high temporal divergence, indicating that prompt-level guidance alone cannot fully enforce the fine-grained timing structure required for realistic 24-hour schedule synthesis.

By contrast, ActivityEditor variants achieve the strongest overall performance. In particular, GRPO_Qwen obtains the highest accuracy across all four states, reaching 0.761 on California, 0.768 on Arizona, 0.736 on Georgia, and 0.743 on Oklahoma. It also achieves the lowest activity-type divergence across all four states, reducing Atype to 0.094, 0.101, 0.119, and 0.116, respectively. These results show that the proposed generate-then-edit framework improves not only activity-label prediction but also population-level semantic consistency.

The temporal results further reveal the complementary roles of SFT and GRPO. SFT_Qwen achieves the best Mint on Arizona, Georgia, and Oklahoma, while GRPO_Qwen achieves the best Mint on California and consistently delivers the best accuracy and activity-type consistency. Overall, ActivityEditor better balances sequence fidelity, temporal alignment, and distributional consistency than existing baselines, while requiring only non-spatial user profiles and no target-state training schedules.

5.3 Zero-shot Transferability Analysis

Cold-start remains a fundamental challenge in urban mobility modeling, as unseen regions often lack sufficient historical data. To test whether our framework learns transferable mobility patterns rather than state-specific shortcuts, we conduct cross-state zero-shot transfer experiments. Models trained on California are directly tested on California, Arizona, Georgia, and Oklahoma without domain adaptation, and we consider the reverse setting of Arizona → California. As shown in Table 2, our method outperforms task-specific baselines such as CoPB (Shao et al., 2024) and RAGHome (Liu et al., 2024), while maintaining stable macro-level behavior statistics across target states.

These results show that our rule-guided framework captures transferable non-spatial activity regularities instead of memorizing state-specific schedule patterns. With California as the source domain, performance drops on other states remain limited,

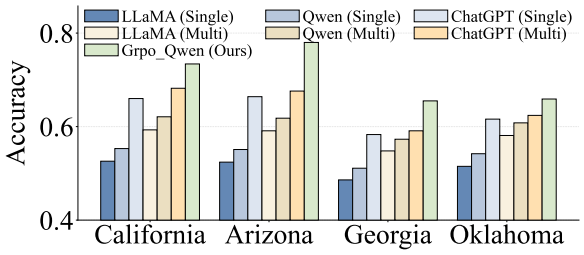

Figure 2: Architecture ablation between single-agent and multi-agent generation modes.

Variant	Acc ↑	Mint ↓	Atype ↓
Agent component ablation			
w/o Intention Agent	0.539	0.586	0.264
w/o Editor Agent	0.681	0.421	0.166
Teacher-supervision ablation			
w/o Teacher CoT	0.697	0.336	0.154
GRPO w/o SFT Teacher	0.716	0.281	0.137
ActivityEditor	**0.752**	**0.217**	**0.108**

Table 3: Ablation study on agent components and teacher-supervision strategies.

and GRPO_Qwen delivers the best overall results in most settings. Arizona → California performs worse than California → Arizona, likely because Arizona offers fewer training samples for learning transferable decision patterns. Even so, the relative ranking of model variants remains largely unchanged, demonstrating the robustness of our framework under data-scarce source domains.

5.4 Ablation Study

We conduct ablation studies to examine whether the proposed gains come from the multi-agent architecture, the two agent components, teacher-generated traces, or GRPO-based rule optimization.

Single-agent vs. multi-agent generation. We first compare general LLMs under single-pass and multi-agent settings. In the single-pass setting, the model directly generates a full 24-hour schedule from the profile; in the multi-agent setting, the Intention Agent first produces a coarse activity plan and the Editor Agent then refines it.

As shown in Figure 2, the multi-agent setting consistently outperforms single-pass generation across ChatGPT, Qwen, and LLaMA. This confirms that daily activity-chain synthesis is a long-horizon constrained generation problem: explicit planning and editing help reduce accumulated temporal and logical errors.

Agent components and teacher supervision. Table 3 further isolates the role of each agent and the effect of GPT-generated teacher traces. Removing either agent reduces performance, with a larger drop from removing the Intention Agent, showing that profile-conditioned planning is important. The no-teacher variants also improve over the model without the Editor Agent, indicating that the gains are not solely from teacher-generated CoT. However, both no-teacher variants remain weaker than the full model, suggesting that teacher traces provide an effective cold-start editing policy, while GRPO further strengthens constraint-aware optimization.

5.5 Training Strategy Ablation

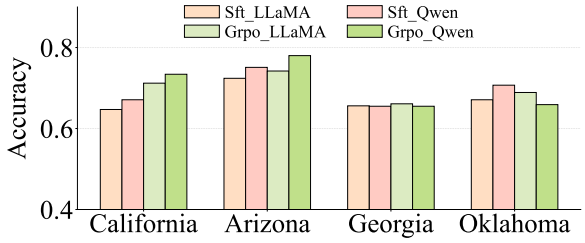

Figure 3: Training strategy ablation between SFT and GRPO.

We further compare SFT and GRPO to analyze the effects of supervised imitation and rule-guided optimization. As shown in Figure 3, SFT mainly provides distributional alignment, while GRPO further improves constraint-aware schedule refinement through rule-based rewards.

6 Conclusion

We presented ActivityEditor, a dual-agent framework for zero-shot, profile-conditioned activity-chain synthesis in cross-state scenarios. By combining intention-driven draft generation with rule-constrained iterative refinement, the framework learns transferable schedule-level behavioral regularities rather than memorizing state-specific patterns. Experimental results on the NHTS dataset show that ActivityEditor substantially outperforms existing baselines in statistical fidelity, temporal alignment, and distributional consistency across four US states. Future work will explore extending the framework to finer temporal resolutions, integrating spatial context for POI-level generation, and scaling to larger population simulations.

Limitations

Despite its promising performance, several limitations of ACTIVITYEDITOR warrant further investigation:

- **Non-Spatial Scope:** The current framework synthesizes activity chains defined solely by activity type and temporal extent, without modeling spatial locations. It therefore cannot produce POI-level or coordinate-level outputs needed for micro-level urban analysis. Future research will explore integrating external geographic context (e.g., POI databases, land-use maps) to extend the framework toward spatially-grounded schedule generation.

- **Computational Efficiency:** The multi-turn iterative refinement between the Intention-driven and Editor agents incurs significant computational overhead, which may limit its scalability for massive population simulations. Knowledge distillation into smaller language models or the optimization of reinforcement learning efficiency represents a viable direction to reduce inference latency.

- **Robustness and Hallucination:** As an LLM-based framework, the output of ACTIVITYEDITOR is not fully controllable. While we utilize a structured parser to extract the expected activity chains, it may occasionally fail to parse non-conforming outputs. Additionally, the agent may hallucinate activity types outside the predefined category set or generate temporally implausible durations. While rule-based reward signals mitigate such issues during training, ensuring absolute robustness in open-world deployment remains a challenge.

Ethical Considerations

All human mobility data used in the experiments come from publicly available open-source datasets (McGuckin and Fucci, 2018). We do not attempt to extract any personal information from these datasets.

AI Usage

We state that AI-assisted technologies were used strictly for the purpose of language polishing and grammar correction to enhance the clarity of our writing. The authors take full responsibility for the integrity of the work, including the validity of all experimental results and the accuracy of all references.

References

Prabin Bhandari, Antonios Anastasopoulos, and Dieter Pfoser. 2024. Urban mobility assessment using llms. In *Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems*, pages 67–79.

John L Bowman and Moshe E Ben-Akiva. 2001. Activity-based disaggregate travel demand model system with activity schedules. *Transportation research part a: policy and practice*, 35(1):1–28.

Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3(1):1376.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025a. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Jie Feng, Yuwei Du, Jie Zhao, and Yong Li. 2025b. Agentmove: A large language model based agentic framework for zero-shot next location prediction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1322–1338.

Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 world wide web conference*, pages 1459–1468.

Jie Feng, Zeyu Yang, Fengli Xu, Haisu Yu, Mudan Wang, and Yong Li. 2020. Learning to simulate human mobility. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3426–3433.

Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying human mobility via trajectory embeddings. In *IJCAI*, volume 17, pages 1689–1695.

Konstadinos G Goulias, Chandra R Bhat, Ram M Pendyala, Yali Chen, Rajesh Paleti, Karthik C Konduri, Guoxiong Huang, and Hsi-Hwa Hu. 2011. Simulator of activities, greenhouse emissions, networks, and travel (simagent) in southern california: Design, implementation, preliminary findings, and integration plans. In *2011 IEEE Forum on Integrated and Sustainable Transportation Systems*, pages 164–169. IEEE.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Siyu Li, Toan Tran, Haowen Lin, John Krumm, Cyrus Shahabi, Lingyi Zhao, Khurram Shafique, and Li Xiong. 2024. Geo-llama: Leveraging llms for human mobility trajectory generation with spatiotemporal constraints. *arXiv preprint arXiv:2408.13918*.

Xishun Liao, Haoxuan Ma, Yifan Liu, Yuxiang Wei, Brian Yueshuai He, Chris Stanford, and Jiaqi Ma. 2025. Next-generation travel demand modeling with a generative framework for household activity coordination. *arXiv preprint arXiv:2507.08871*.

Peijie Liu, Fengli Xu, and Yong Li. 2026. Travelreasoner: Leveraging large reasoning models to address mobility data gap. In *Proceedings of the ACM Web Conference 2026*, pages 9341–9351.

Yifan Liu, Xishun Liao, Haoxuan Ma, Brian Yueshuai He, Chris Stanford, and Jiaqi Ma. 2024. Human mobility modeling with household coordination activities under limited information via retrieval-augmented llms. *arXiv preprint arXiv:2409.17495*.

Nancy McGuckin and Anthony Fucci. 2018. Summary of travel trends: 2017 national household travel survey.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, and 1 others. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Chenyang Shao, Fengli Xu, Bingbing Fan, Jingtao Ding, Yuan Yuan, Meng Wang, and Yong Li. 2024. Chain-of-planned-behaviour workflow elicits few-shot mobility generation in llms. *arXiv preprint arXiv:2402.09836*.

Ke Sun, Tieyun Qian, Tong Chen, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2020. Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 214–221.

Jiawei Wang, Renhe Jiang, Chuang Yang, Zengqing Wu, Makoto Onizuka, Ryosuke Shibasaki, Noboru Koshizuka, and Chuan Xiao. 2024a. Large language models as urban residents: An llm agent framework for personal mobility generation. *Advances in Neural Information Processing Systems*, 37:124547–124574.

Xinglei Wang, Meng Fang, Zichao Zeng, and Tao Cheng. 2023. Where would i go next? large language models as human mobility predictors. *arXiv preprint arXiv:2308.15197*.

Yu Wang, Tongya Zheng, Yuxuan Liang, Shunyu Liu, and Mingli Song. 2024b. Cola: Cross-city mobility transformer for human trajectory simulation. In *Proceedings of the ACM Web Conference 2024*, pages 3509–3520.

Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and 1 others. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level reward design. *arXiv preprint arXiv:2505.11821*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, and 1 others. 2025. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.

Yu Zheng, Yuming Lin, Liang Zhao, Tinghai Wu, Depeng Jin, and Yong Li. 2023. Spatial planning of urban communities via deep reinforcement learning. *Nature Computational Science*, 3(9):748–762.

A Appendix

A.1 Dataset Details and Leakage Prevention

A.1.1 Dataset Scope and Cross-State Heterogeneity

We use the 2017 National Household Travel Survey (NHTS), which provides socio-demographic profiles paired with detailed 24-hour activity and travel diaries. Although our experiments are based on one unified data source, NHTS covers diverse states, household types, and socio-demographic groups. It is therefore suitable for evaluating cross-state transfer under a consistent survey framework, rather than being a single-city or single-region benchmark.

We extract four state-level subsets: California (CA), Arizona (AZ), Georgia (GA), and Oklahoma (OK). California is used as the source state, while Arizona, Georgia, and Oklahoma are treated as unseen zero-shot target states. These states differ substantially in data scale and regional characteristics. For example, California contains over 56,000 home-activity records, while Arizona and Oklahoma contain approximately 6,900 and 3,100 home-activity records, respectively. This imbalance reflects a realistic cold-start setting where some regions lack sufficient historical activity schedules for training.

We emphasize that our experiments evaluate cross-state zero-shot transfer within the NHTS framework, rather than cross-dataset generalization. This scope avoids overstating the empirical claim while still testing whether profile-conditioned activity regularities learned from a data-rich source state can transfer to unseen target states.

A.1.2 Cross-State Activity Regularities

The NHTS subsets exhibit shared high-level activity regularities across states. As shown in Figure 5, activities such as home, work, and shopping consistently dominate daily routines. Figure 7 further shows that the distribution of activity-chain lengths follows a similar right-skewed pattern across states, typically peaking between three and five activities per person.

The dataset also reflects stable temporal constraints. Figure 6 shows that long-duration activities such as staying at home or working usually span hundreds of minutes, while short transitional activities such as drop-off/pick-up remain much shorter. These observations motivate our rule-based reward design, which encourages full-day coverage,

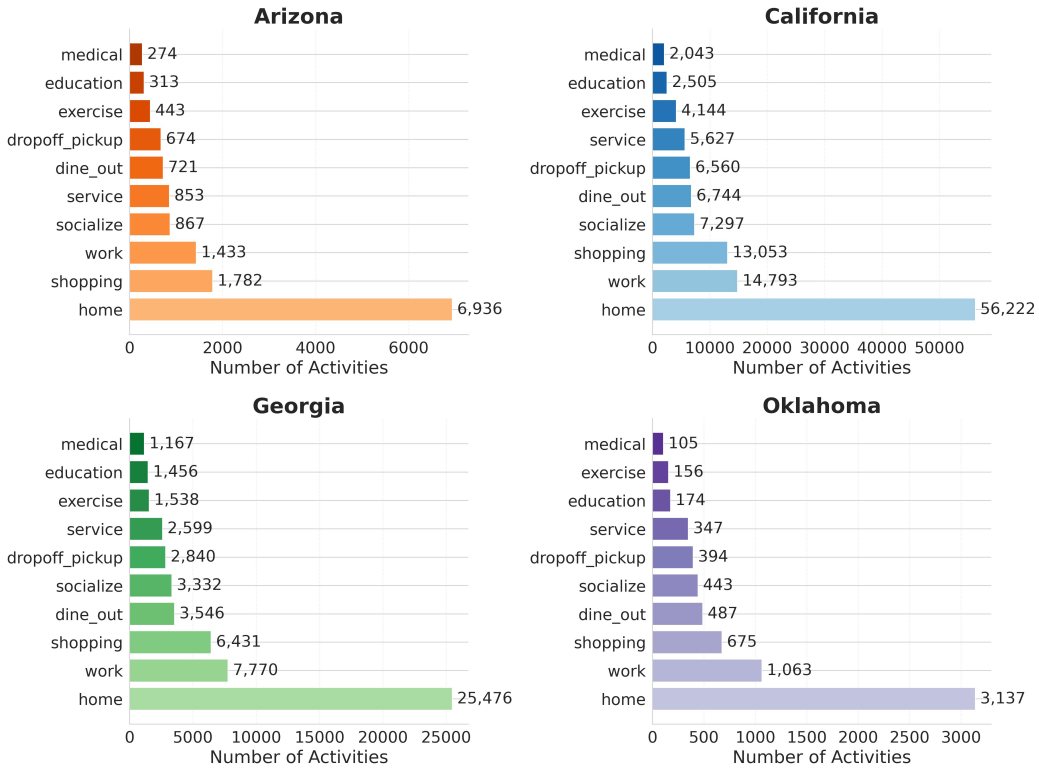

Figure 4: Activity number distribution by state.

temporal continuity, realistic durations, and valid activity transitions without requiring target-state training schedules.

A.1.3 Train/Validation/Test Split

For the California source state, we split users into training, validation, and test sets with an 80/10/10 ratio. The California subset contains 22,082 users, of which 17,666 are used for training, 2,208 for validation, and 2,208 for testing. The California evaluation subset used in the main experiments is sampled only from the held-out California test split, and no user in this subset appears in training.

Arizona, Georgia, and Oklahoma are used only as zero-shot target states. Their activity schedules are never used for training, validation, retrieval, prompt construction, teacher-trace generation, GRPO reward estimation, or hyperparameter tuning.

A.1.4 Leakage Prevention Protocol

To address the concern that GPT-generated chain-of-thought supervision may introduce data leakage, we enforce strict data isolation throughout the pipeline. The key principle is that held-out test labels and target-state schedules are never accessed before final evaluation.

Dataset partitioning is performed before teacher-trace generation. GPT-5.2 is used only to generate editing traces for users in the California training split. Teacher traces are never generated for California validation users, California test users, or any users from Arizona, Georgia, or Oklahoma. Thus, the teacher model observes only training instances and never observes the labels used for final evaluation.

During SFT, the Editor Agent is trained only on teacher traces derived from California training users. During GRPO, rewards are computed using programmatic rules and reference statistics estimated only from the California training split. Target-state labels are not accessed during reward construction. Evaluation is performed after training is complete and uses only generated schedules and held-out ground-truth schedules.

A.1.5 Train–Test Overlap Audit

We further audit potential overlap between training and test instances. We check whether training and test splits contain identical user profiles, identical complete activity chains, or highly similar partial activity sequences. The audit shows that no complete test schedule appears in the training split. Some partial activity-sequence overlap natu-

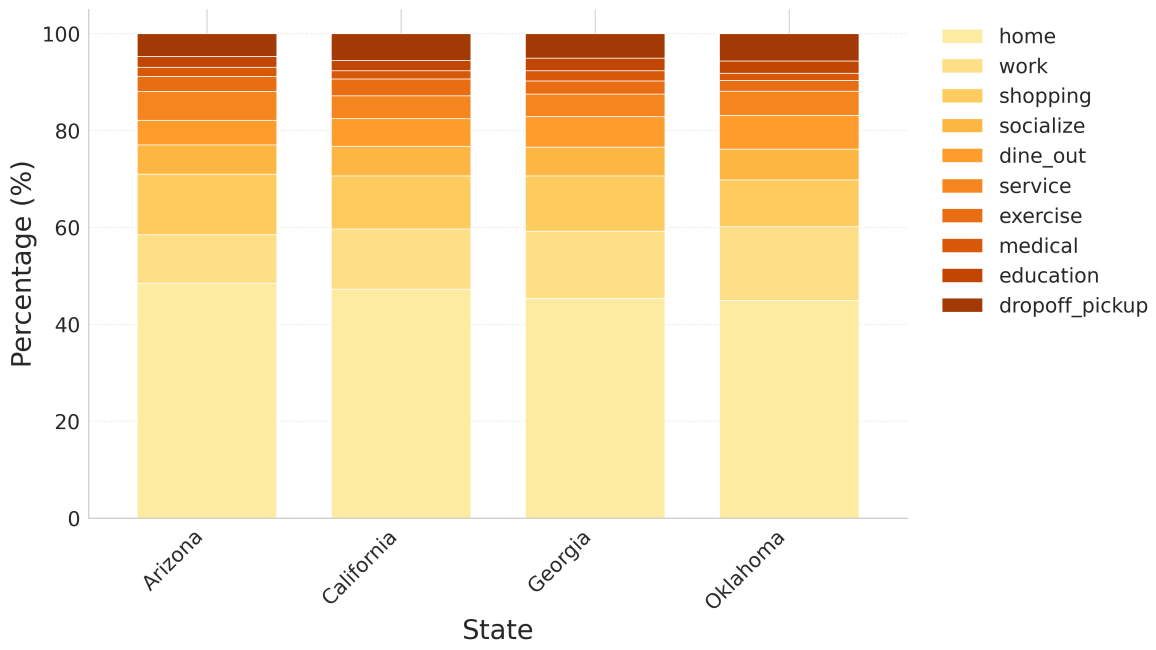

Figure 5: Activity type distribution by state.

Stage	Data used	Access to held-out or target labels
Teacher-trace generation	California train only	No
SFT training	California train teacher traces	No
GRPO training	California train statistics and rules	No
Hyperparameter tuning	California validation only	No target-state labels
Final evaluation	California test / AZ / GA / OK	Evaluation only

Table 4: Data access protocol across training and evaluation stages.

Audit criterion	Train–Train	Train–Test
Identical user profiles	1,847 (10.5%)	0 (0.0%)
Identical activity chains	924 (4.2%)	0 (0.0%)
At least 50% activity-sequence overlap	3,612 (20.4%)	1,574 (8.9%)
Duplicated complete test schedules	–	0/50 (0.0%)

Table 5: Training–test leakage audit on the California subset.

rally exists because many daily routines share common patterns, such as home–work–home or home–shopping–home, but this reflects recurring human behavior rather than duplicated user instances.

The average activity-type and duration distribution overlap between the California training and test splits is measured by Jensen–Shannon divergence, with JSD = 0.042. This indicates that the training and test distributions are behaviorally related, as expected for users from the same state, while remaining instance-level separated.

A.2 Detailed Description of Baselines

To comprehensively evaluate the proposed framework, we compare ActivityEditor with four groups of baselines: traditional sequence models, prompt-based and retrieval-augmented methods, recent activity-time LLM baselines, and general zero-shot LLMs. All baselines are evaluated under the same non-spatial activity-chain synthesis setting whenever applicable.

Traditional sequence baselines. DeepMove (Feng et al., 2018): DeepMove is a classic human mobility prediction model based on attentional recurrent neural networks. It models sequential transition patterns and multi-level periodic behaviors from users' historical trajectories. In our setting, it serves as a representative traditional sequence model that relies on historical mobility patterns.

LSTPM (Sun et al., 2020): LSTPM captures both long-term and short-term user mobility preferences by combining a context-aware non-local network with a geo-dilated recurrent neural network. It is originally designed for next-location prediction, and we adapt it as another sequence-

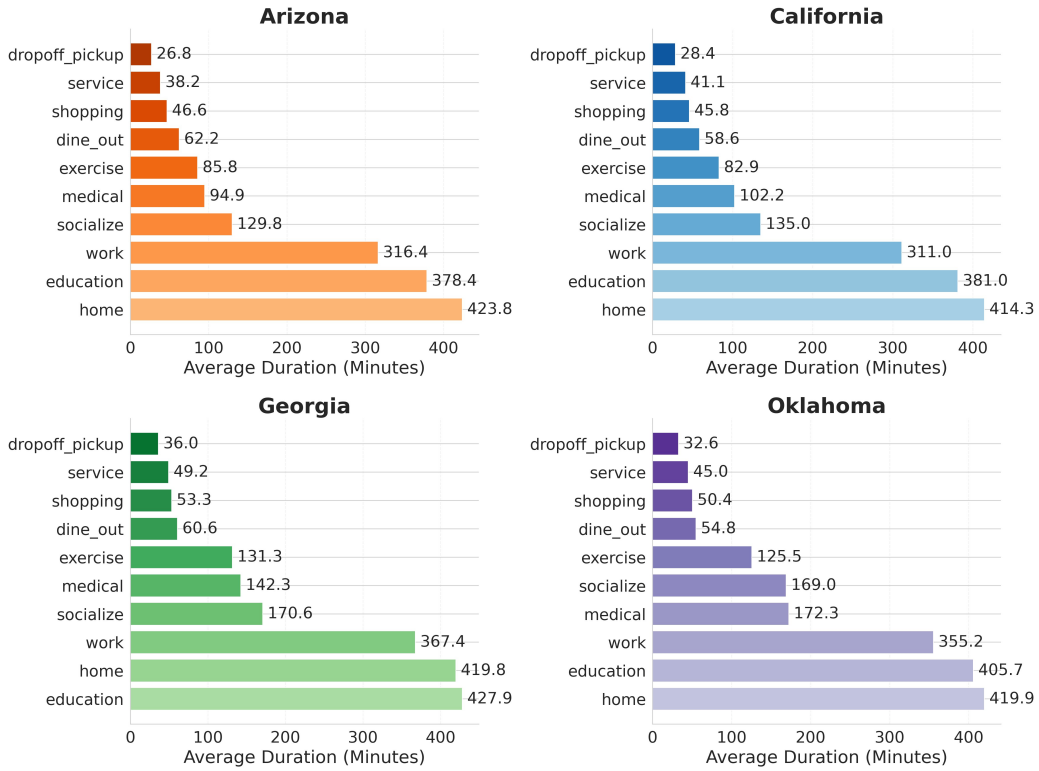

Figure 6: Activity duration by state.

based baseline for activity-chain generation.

Prompt-based and retrieval-augmented baselines. CoPB (Shao et al., 2024): CoPB is grounded in the Theory of Planned Behavior and uses a structured prompt workflow to guide LLMs to reason about users' beliefs, intentions, and behavioral constraints before generating daily activity sequences. It represents a prompt-based LLM baseline for mobility generation.

RAGHome (Liu et al., 2024): RAGHome introduces a retrieval-augmented generation mechanism that retrieves historical mobility records of individuals with similar characteristics and incorporates them as contextual examples for the LLM. It serves as a retrieval-enhanced baseline for profile-conditioned schedule generation.

Recent activity-time LLM baselines. Bhandari24 (Bhandari et al., 2024): Bhandari24 studies the use of LLMs for urban mobility assessment and activity-related reasoning. We adapt its activity-time generation strategy to our non-spatial setting by using the same socio-demographic profile inputs and asking the model to generate a complete 24-hour activity schedule.

TravelReasoner (Liu et al., 2026): TravelReasoner leverages large reasoning models to address mobility data gaps through reasoning-aligned activity generation. We include it as a strong LLM-based activity-time baseline. For fair comparison, we implement it with the same Qwen3-8B backbone and evaluate it under the same profile-conditioned input setting.

LLMob (Wang et al., 2024a): LLMob frames LLMs as urban residents and uses agent-style reasoning to simulate personal mobility behavior. We adapt it to generate non-spatial daily activity schedules from socio-demographic profiles. Unlike ActivityEditor, LLMob follows a primarily single-pass generation paradigm and does not include an explicit iterative editing stage.

General zero-shot LLM baselines. ChatGPT-5.2: ChatGPT-5.2 is evaluated as a proprietary general-purpose LLM baseline. We prompt it directly to generate a complete 24-hour activity schedule from the given socio-demographic profile. We report both single-pass generation and dual-agent prompting variants.

Qwen3-8B: Qwen3-8B is used as an open-source LLM baseline. In the zero-shot setting, it generates schedules directly from user profiles without task-specific fine-tuning. It also serves as

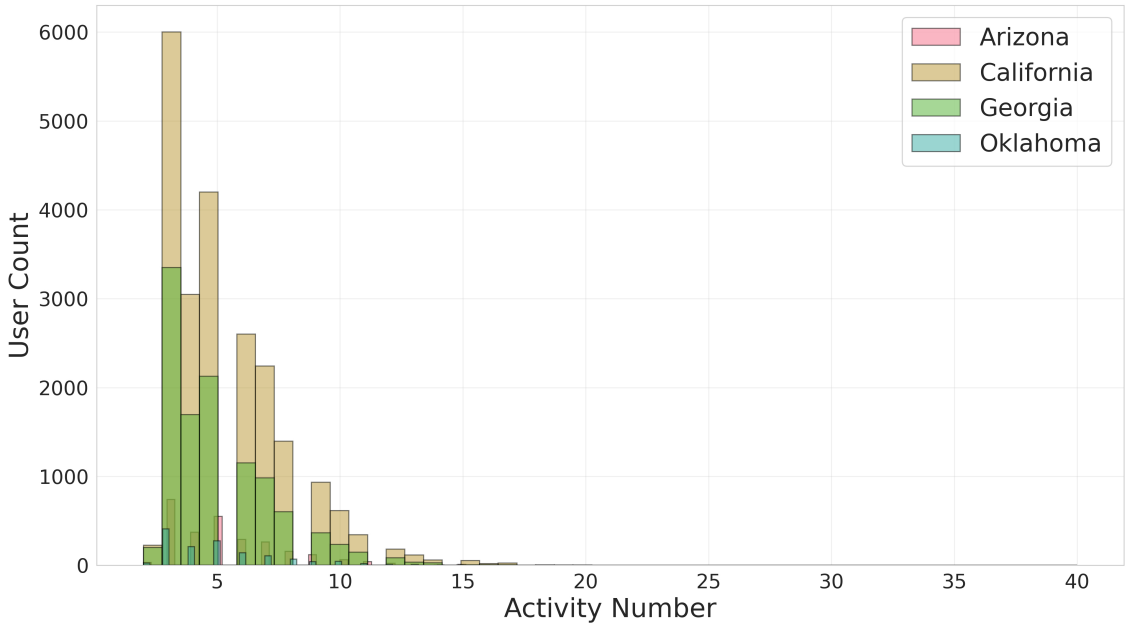

Figure 7: Chain length distribution by state.

the backbone for our SFT and GRPO-based ActivityEditor variants.

LLaMA3.1-8B: LLaMA3.1-8B is included as another open-source general-purpose LLM baseline. Similar to Qwen3-8B, we evaluate it under both single-pass and dual-agent prompting settings to examine whether the proposed Generate-then-Edit structure generalizes across different LLM backbones.

A.3 Prompt Examples

For completeness, this subsection summarizes the core prompts used throughout our activity-chain synthesis framework.

A.3.1 Prompt of Intention-driven Agent

```
You are a daily schedule planner. Given a
    person's profile, generate a complete
    and realistic 24-hour schedule with
    exact start/end times.

RULES:
- Use ONLY these activity types: home, work,
     education, shopping, service, medical,
    dine_out, socialize, exercise,
    dropoff_pickup
- Schedule MUST cover full 24 hours: start
    at 00:00, end at 24:00, NO gaps or
    overlaps
- Must start and end with "home"
- Consecutive activities must connect (
    end_time of one = start_time of the next
    )
- Durations must be realistic based on the
    person's profile

OUTPUT FORMAT:
{
  "reasoning": "Brief explanation of the
     schedule choices",
  "schedule": [
    {"activity": "home", "start_time":
     "00:00", "end_time": "07:30"},
    {"activity": "work", "start_time":
     "07:30", "end_time": "17:00"},
    {"activity": "home", "start_time":
     "17:00", "end_time": "24:00"}
  ]
}

USER MESSAGE FORMAT:
Generate a complete 24-hour schedule for
    this person:

- Age range: {age_range}
- Gender: {gender}
- Race: {race}
- Education: {education}
- Employment status: {employment_status}
- Work schedule: {work_schedule}
- Occupation: {occupation}
- Primary activity: {primary_activity}
- Work from home: {work_from_home}
- Driver on travel day: {
    driver_on_travel_day}
- Distance to work (miles): {
    distance_to_work_miles}
- Work state: {work_state}
```

A.3.2 Prompt of Editor Agent

```
You are an Editor Agent for daily schedule
    refinement.

Your task:
1. Check the INITIAL SCHEDULE against 5
    constraint types:
```

```
   Physical, Logical, Common Sense, Temporal,
     and Coherence
2. Identify any violations in the draft
3. Apply edit operations from the action
    space:
   ADD / DELETE / SHIFT / REPLACE
4. Output the final constraint-satisfying
    schedule

OUTPUT FORMAT:
[THOUGHT]
Constraint Checking:
1. Physical Constraints (Hard): overlaps, 24
    h coverage, ends at 24:00
2. Logical Constraints (Hard): starts/ends
    at home, starts at 00:00, consecutive
    identical activities must be merged
3. Common Sense Constraints (Soft):
    activities match socio-demographic
    profile
4. Temporal Constraints (Soft): realistic
    durations for each activity type
5. Coherence Constraints (Soft): logical
    transitions, not over-fragmented

Edit Operations Applied:
- ADD: activity 'Y' at time HH:MM (reason)
- DELETE: activity 'X' at index N (reason)
- SHIFT: activity 'Z' start/end time
    adjustment (reason)
- REPLACE: activity 'A' -> 'B' (reason)

Final Result: All constraints satisfied /
    Violations remain
[/THOUGHT]

[JSON]
[final schedule as JSON array]
[/JSON]

USER MESSAGE FORMAT:
PERSON PROFILE:
{user_profile as JSON}

INITIAL SCHEDULE:
{initial_schedule as JSON}
```

A.3.3 Prompt of Data Generation Teacher Model

```
SYSTEM PROMPT:
You are a schedule planning expert. You
    understand human behavior patterns and
    generate realistic daily schedules.

USER MESSAGE FORMAT:
You are a Critic and Editor Agent. Your job
    is to validate and refine a daily
    schedule.

PERSON PROFILE:
{user_profile as JSON}

INITIAL SCHEDULE (from unified 2-stage
    generator):
{initial_schedule as JSON}

GROUND TRUTH REFERENCE (for comparison):
{ground_truth_schedule as JSON}

YOUR TASK:
Act as the Editor. Check constraints on the
    INITIAL SCHEDULE, identify violations,
    and apply edits to match the GROUND
    TRUTH.

OUTPUT FORMAT:
[THOUGHT]
Constraint Checking:
1. Physical (Hard): overlaps? 24h coverage?
    ends at 24:00? -> Yes / No
2. Logical (Hard): starts/ends home? starts
    at 00:00? -> Yes / No
3. Common Sense (Soft): activities match
    profile? -> Yes / No
4. Temporal (Soft): realistic durations? ->
    Yes / No
5. Coherence (Soft): logical flow? not
    fragmented? -> Yes / No

Edits to Match Ground Truth:
If already matches: No edits needed
Otherwise list each operation:
- DELETE: 'X' at idx N (reason)
- ADD: 'Y' at time HH:MM (reason)
- SHIFT: 'Z' time adjustment (reason)
- REPLACE: 'A' -> 'B' (reason)

Final Result:
All constraints satisfied after edits? Yes /
     No
[/THOUGHT]

[JSON]
[final schedule - must match the GROUND
    TRUTH]
[/JSON]

Be thorough in checking each constraint on
    the INITIAL SCHEDULE.
Show edit operations explicitly if
    violations are found.
```

A.3.4 Prompt of SFT Student Model

```
You are a Critic and Editor Agent for daily
    schedule refinement.

Your task:
1. Check the INITIAL SCHEDULE against 5
    constraint types:
   Physical, Logical, Common Sense, Temporal,
     and Coherence
2. Identify any violations
3. Apply edit operations:
   ADD / DELETE / SHIFT / REPLACE
4. Output the refined schedule

OUTPUT FORMAT:
[THOUGHT]
Constraint Checking:
1. Physical Constraints (Hard): overlaps, 24
    h coverage, ends at 24:00
2. Logical Constraints (Hard): starts/ends
    at home, starts at 00:00
3. Common Sense Constraints (Soft): age/
    employment appropriate activities
```

```
4. Temporal Constraints (Soft): realistic
    durations
5. Coherence Constraints (Soft): logical
    transitions, not over-fragmented

Applying Edit Operations:
- DELETE: activity 'X' at index N (reason)
- ADD: activity 'Y' at time HH:MM (reason)
- SHIFT: activity time adjustment (reason)
- REPLACE: activity type change (reason)

Final Result: Yes / No
[/THOUGHT]

[JSON]
[refined schedule as JSON array]
[/JSON]

USER MESSAGE FORMAT:
PERSON PROFILE:
{user_profile as JSON}

INITIAL SCHEDULE:
{initial_schedule as JSON}

TRAINING TARGET:
[THOUGHT]
{teacher-generated constraint-checking trace
     and edit operations}
[/THOUGHT]

[JSON]
{final refined schedule matching ground
    truth}
[/JSON]
```

A.3.5 Prompt of GRPO Training

```
You are a Critic and Editor Agent for daily
    schedule refinement. Your task is to
    check and refine an initial schedule
    based on the person's profile.

Your task:
1. Check the INITIAL SCHEDULE against 5
    constraint types:
   Physical, Logical, Common Sense, Temporal,
     and Coherence
2. Identify any violations
3. Apply edit operations:
   ADD / DELETE / SHIFT / REPLACE
4. Output the refined schedule

OUTPUT FORMAT:
[THOUGHT]
Constraint Checking and Edit Operations
    applied to the initial schedule.
[/THOUGHT]

[JSON]
[refined schedule as JSON array]
[/JSON]

The schedule must start at 00:00 and end at
    24:00, covering the full day without
    gaps or overlaps.

USER MESSAGE FORMAT:
PERSON PROFILE:
{user_profile as JSON}

INITIAL SCHEDULE:
{initial_schedule from Intention-Driven
    Agent as JSON}
```

A.4 Detailed Parameter Settings

To ensure the reproducibility of our experiments, we provide the complete hyperparameter configurations for both the Supervised Fine-Tuning (SFT) and Group Relative Policy Optimization (GRPO) stages in Table 4.

Hyperparameter	SFT Stage
Base Model	Qwen3-8B
LoRA Rank (r)	32
LoRA Alpha	64
LoRA Dropout	0.05
Training Epochs	3 epochs
Peak Learning Rate	5e-5
Learning Rate Scheduler	Cosine
Warmup	50 steps
Per-device Batch Size	2
Gradient Accumulation Steps	8
Max Sequence Length	4096

Table 6: Hyperparameter settings for the SFT stage.

Hyperparameter	GRPO Stage
Base Model	Qwen3-8B
LoRA Rank (r)	32
LoRA Alpha	64
LoRA Dropout	0.05
Training Steps	1000 steps
Peak Learning Rate	1e-5
Learning Rate Scheduler	Cosine
Warmup	0.1 ratio
Gradient Accumulation Steps	4
Max Sequence Length	4096
Roll-out Samples per Prompt	8
Sampling Temperature	0.7
Min_p	0.1

Table 7: Hyperparameter settings for the GRPO stage.

All models were trained using the bfloat16 precision format to optimize memory usage while maintaining numerical stability. The AdamW optimizer was utilized across all training phases, with the beta1 and beta2 parameters set to 0.9 and 0.99, respectively, alongside a weight decay coefficient of 0.1. To prevent gradient explosion, a gradient clipping threshold of 1.0 was universally applied across all parameter updates. As stated in the main text, both stages utilized the exact same Low-Rank Adaptation configuration to efficiently update the attention and feed-forward modules. The specific trainable adapters were applied to the q_proj,

Method	Acc ↑	Mint ↓	Atype ↓
LLMob	0.643 ± 0.031	0.479 ± 0.025	0.217 ± 0.022
CoPB	0.703 ± 0.024	0.577 ± 0.028	0.141 ± 0.019
SFT-Qwen	0.753 ± 0.020	0.231 ± 0.035	0.104 ± 0.018
GRPO-Qwen	$\mathbf{0.761 \pm 0.018}$	$\mathbf{0.202 \pm 0.031}$	$\mathbf{0.094 \pm 0.015}$

Table 8: Robustness analysis on the held-out California test subset with 95% bootstrap confidence intervals. We perform 1,000 bootstrap resampling trials.

k_proj, v_proj, o_proj, gate_proj, up_proj, and down_proj layers. The dataset was split into training, validation, and testing sets with a ratio of 8:1:1, and all reported results were averaged over three independent runs.

A.5 Robustness and Statistical Significance

We evaluate the robustness of the main results using bootstrap confidence intervals on the held-out California test subset. Specifically, we perform 1,000 bootstrap resampling trials and report 95% confidence intervals for the main evaluation metrics. We also conduct paired bootstrap tests between ActivityEditor and the strongest competing baselines to assess whether the observed improvements are statistically reliable.

As shown in Table 8, GRPO-Qwen achieves the best mean performance on all three metrics and also exhibits relatively narrow confidence intervals. Compared with the strongest non-ActivityEditor baseline, CoPB, GRPO-Qwen improves Acc from 0.703 to 0.761, reduces Mint from 0.577 to 0.202, and reduces Atype from 0.141 to 0.094.

The paired bootstrap tests show that the improvements of GRPO-Qwen over CoPB and LLMob are statistically significant on the main metrics ($p < 0.01$). These results indicate that the observed gains are not due to random variation in the sampled test subset. The improvement is especially clear for temporal interval divergence and activity-type distribution divergence, supporting our claim that ActivityEditor improves constrained schedule synthesis rather than merely increasing slot-level activity accuracy.

A.6 Evaluation Metrics Formulation

Before computing the metrics, each continuous daily schedule is discretized into a fixed-length sequence of 96 time slots, representing 15-minute intervals. The assigned activity for each slot is the one with the longest temporal overlap. Let $G \in \mathbb{Z}^{N\times 96}$ and $T \in \mathbb{Z}^{N\times 96}$ denote the generated and ground truth slot matrices for N users, respectively. We group the 12 metrics into three mathematical categories.

Sequence Similarity Metrics. These metrics evaluate how closely the generated activity sequence matches the ground-truth sequence at the token and structural levels.

Accuracy (Acc) calculates the slot-level exact match rate between the generated and ground truth sequences:

$$\text{Acc} = \frac{1}{N \times 96} \sum_{i=1}^{N} \sum_{t=1}^{96} \mathbf{1}[G_{i,t} = T_{i,t}] \quad (4)$$

Macro F1-Score computes the balanced accuracy over $C = 11$ activity classes:

$$\text{F1}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^{K} \frac{2 \cdot \text{Prec}_k \cdot \text{Recall}_k}{\text{Prec}_k + \text{Recall}_k} \quad (5)$$

Edit Distance (edit_dist) calculates the minimum edit cost, normalized by the sequence length:

$$\text{edit_dist} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{Levenshtein}(G_i, T_i)}{96} \quad (6)$$

BLEU Score applies n-gram sequence overlap precision up to 4-gram:

$$\text{BLEU} = \frac{1}{N} \sum_{i=1}^{N} \text{sentence_BLEU}(T_i, G_i) \quad (7)$$

Temporal Alignment Metrics. These metrics measure whether the generated schedules preserve realistic temporal patterns, including activity durations and starting times.

These metrics utilize the Jensen-Shannon Divergence (JSD) to evaluate duration and start-time distributions.

Macro_int evaluates the marginal distribution of continuous activity durations (run lengths), denoted as frequency vector m:

$$\text{macro_int} = \text{JSD}\left(\frac{m^{\text{gt}}}{\|m^{\text{gt}}\|_1} \middle\| \frac{m^{\text{gen}}}{\|m^{\text{gen}}\|_1} \right) \quad (8)$$

Micro_int computes the divergence on the flattened joint distribution matrix M of activity types and durations:

$$\text{micro_int} = \text{JSD}\left(\frac{M^{\text{gt}}_{\text{flat}}}{\|M^{\text{gt}}\|_1} \middle\| \frac{M^{\text{gen}}_{\text{flat}}}{\|M^{\text{gen}}\|_1} \right) \quad (9)$$

Initial Draft	Reasoning Cues	Edited Output	Ground Truth
[00:00–07:45] Home		[00:00–07:45] Home	[00:00–07:45] Home
[07:45–16:45] Work	occupation → Evening Work	[07:45–16:30] Work	[07:45–16:30] Work
[16:45–18:00] Home	relationship → Shopping	[16:30–17:20] Shopping	[16:30–17:30] Shopping
[17:45–19:00] Work	constraint → Fix overlap	[17:20–19:50] Work	[17:30–17:45] Service
[19:00–19:35] Shopping	short commute → no extra Home stay	[19:50–24:00] Home	[17:45–20:30] Work
[19:35–24:00] Home			[20:30–24:00] Home
Status: invalid	**Signals:** profile + constraint check	**Status:** valid	**Status:** observed

Table 9: Case study of profile-grounded trajectory editing.

Macro_hour measures the divergence of the marginal distribution of activity onset times (start slots), denoted as vector h:

$$\text{macro_hour} = \text{JSD}\left(\frac{h^{\text{gt}}}{\|h^{\text{gt}}\|_1} \middle\| \frac{h^{\text{gen}}}{\|h^{\text{gen}}\|_1}\right) \quad (10)$$

Micro_hour calculates the divergence on the joint distribution matrix H of activity types and their onset times:

$$\text{micro_hour} = \text{JSD}\left(\frac{H^{\text{gt}}_{\text{flat}}}{\|H^{\text{gt}}\|_1} \middle\| \frac{H^{\text{gen}}_{\text{flat}}}{\|H^{\text{gen}}\|_1}\right) \quad (11)$$

Distributional Consistency Metrics. These metrics assess whether the generated population reproduces the global statistical properties and behavioral diversity of the real-world data.

Data_jsd evaluates population-level trajectory diversity by treating each full 96-slot sequence as a single token with probability p:

$$\text{data_jsd} = \text{JSD}(p^{\text{gt}} \| p^{\text{gen}}) \quad (12)$$

Act_type calculates the divergence of the total time budget allocated to each activity category p_k:

$$\text{act_type} = \text{JSD}(p_k^{\text{gt}} \| p_k^{\text{gen}}) \quad (13)$$

Uni_act_type computes the divergence based on the frequency of distinct activity episodes per category q_k:

$$\text{uni_act_type} = \text{JSD}(q_k^{\text{gt}} \| q_k^{\text{gen}}) \quad (14)$$

Traj_len measures the divergence in the distribution of the total number of distinct activity episodes per user, denoted as count vector r:

$$\text{traj_len} = \text{JSD}\left(\frac{r^{\text{gt}}}{\|r^{\text{gt}}\|_1} \middle\| \frac{r^{\text{gen}}}{\|r^{\text{gen}}\|_1}\right) \quad (15)$$

A.7 Case Study: Profile-Grounded Activity-Chain Editing

We present a case study based on a real user from the NHTS California subset. The profile indicates that the user is a female, a mother, employed full-time in a professional, managerial, or technical occupation, does not work from home, drives on the travel day, and has a short commute distance of 4.15 miles. Her primary activity is working.

Reasoning trace. Reasoning process is as follows:

[THOUGHT]
Constraint Checking (5 types):

1. **Physical Constraints (Hard):** - Detected overlap: Home 16:45–18:00 and Work 17:45–19:00
Result: FAIL

2. **Logical Constraints (Hard):** - Starts at Home? Yes - Ends at Home? Yes - Full-day coverage? Yes
Result: PASS

3. **Common Sense Constraints (Soft):** - relationship → Shopping - occupation → Evening Work
Result: informative cues

4. **Temporal Constraints (Soft):** - Shopping is more plausible after daytime Work than after evening Work
Result: needs adjustment

5. **Coherence Constraints (Soft):** - short commute → avoid extra Home stay before returning to Work
Result: needs adjustment

Editing decision: - shift the first Work block earlier, - move Shopping to the interval after daytime Work, - delay the

second Work block until after Shopping, - remove the redundant late Shopping segment.

Output: - Home 00:00–07:45 - Work 07:45–16:30 - Shopping 16:30–17:20 - Work 17:20–19:50 - Home 19:50–24:00

Result: valid schedule with improved profile consistency and closer structural alignment to the observed trajectory. [/THOUGHT]

This case study shows the role of the Editor Agent: improving a profile-informed draft by fixing hard errors and refining activity order with profile-conditioned commonsense. The result is not an exact match to the observed day, but a more realistic and coherent daily activity chain.