Designing A Scalable LLM Agent Framework for Large-scale Urban Segregation Simulation

Qingbin Zeng*
zqb24@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Yuwei Yan*
yyan791@connect.hkust-gz.edu.cn
Hong Kong University of Science and Technology (GuangZhou)
Guangzhou, China

Zhiheng Zhen
zhengzhi24@mails.tsinghua.edu.cn
Shenzhen International Graduate School Tsinghua University
Shenzhen, China

Jingzhe Yuan
yuanjz22@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Jie Feng
fengjie@bza.edu.cn
Zhongguancun Academy
Beijing, China

Jun Zhang
zhangjun990222@qq.com
Tsinghua University
Beijing, China

Fengli Xu†
fenglixu@tsinghua.edu.cn
Tsinghua University
Beijing, China

James Evans
jevans@uchicago.edu
University of Chicago
Chicago, USA

Yong Li†
liyong07@tsinghua.edu.cn
Tsinghua University
Beijing, China

Abstract

Agent-based models (ABMs) have long been employed to explore how individual behaviors aggregate into complex societal phenomena in urban space. The rise of Large Language Models (LLMs) offers a new paradigm for ABM, enabling high-fidelity modeling of human behavior. However, urban dynamics often require extensive agent interactions to emerge, and scaling up LLM agent simulations is limited by high-latency remote LLM inference and high costs. To address this, we propose the OpenCity framework for large-scale LLM agent simulation. We design a novel "Group-and-Distill" strategy that significantly reduces costs without sacrificing behavioral diversity by aggregating agents with shared static characteristics while preserving their unique dynamic states. Moreover, a system-level request scheduler is designed to optimize I/O concurrency and communication bottlenecks. Experiments in six cities globally demonstrate that OpenCity achieves a 600x speedup and a 45% reduction in token usage compared to standard baseline models, while maintaining 96% behavioral consistency with independently invoked agent baselines. Based on OpenCity, we conduct first benchmark test on large-scale social segregation with LLM agents. The results show that OpenCity-accelerated generative agents are effective in reproducing socioeconomic segregation patterns. Furthermore, our scalability analysis demonstrates that the realism of emerging social phenomena increases as the agents scale up, quantitatively validating the necessity of large-scale simulations. Finally, we demonstrate the framework's capability as a policy sandbox through a counterfactual analysis of urban equity. Overall, OpenCity provides a robust and efficient framework that unlocks the potential of LLMs for large-scale, high-fidelity multi-agent systems analysis. Code repo is available at https://anonymous.4open.science/r/Anonymous-OpenCity-42BD.

CCS Concepts

• **Computing methodologies** → **Multi-agent systems**.

Keywords

Generative Modeling, Agent Based Simulation, Large Language Models(LLMs), LLM Agents, Efficiency, Urban Simulation

ACM Reference Format:
Qingbin Zeng, Yuwei Yan, Zhiheng Zhen, Jingzhe Yuan, Jie Feng, Jun Zhang, Fengli Xu, James Evans, and Yong Li. 2018. Designing A Scalable LLM Agent Framework for Large-scale Urban Segregation Simulation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Understanding how individual micro-motives aggregate into macro-level societal phenomena is a cornerstone of computational social science and urban planning. Traditionally, researchers have relied on massive observational data (e.g., GPS trajectories) [15, 28] or rule-based Agent-Based Models (ABMs) [14, 18, 19] to study these dynamics. However, both approaches face fundamental limitations: observational data primarily captures what people do but offers limited insight into the motivations underlying urban activity, while rule-based ABMs often rely on rigid, predefined heuristics that fail to capture the nuanced heterogeneity and cognitive complexity of real-world individuals.

*Both authors contributed equally to this research.
†Corresponding Author

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

The recent advance of Large Language Models (LLMs) offers a transformative paradigm shift that leverages remarkable LLM capabilities of commonsense reasoning and role-playing to simulate human behaviors [16, 29]. Unlike previous rule-based agents, these emerging LLM agents generate far more realistic human behaviors [16, 22], and can also explain their inner motives via prompting techniques like chain-of-thought [26]. This capability holds the promise of a bottom-up research method through reproducing complex urban phenomena (e.g., experienced segregation) by faithfully modeling the diverse decision-making processes of every single resident, rather than merely fitting macroscopic curves.

However, the field still faces a severe **scalability challenge** due to the high computation cost. Emergent urban phenomena are inherently population-level properties that manifest only when thousands of diverse agents interact over extended periods. Simulating LLM agents at this magnitude involves millions of computationally intensive inference calls, creating a dual bottleneck of extreme latency and financial cost. The pioneering work on generative agents was thus limited to simulating a mere 25 LLM agents in a small virtual village [16]. Furthermore, the dynamic evolution of agent memories and perceptions in urban environments [16, 31] necessitates that each agent maintains an independent experience. This prevents the straightforward reuse of simulated behaviors from a small sample of the population [3], as doing so would fundamentally undermine the principle of agent heterogeneity critical for simulating a vibrant and diverse urban population. This bottleneck prevents the researchers from truly harnessing the power of LLMs to study large-scale emergent social dynamics.

To break this barrier, we introduce **OpenCity**, a scalable framework designed to enable city-scale simulation with massive LLM agents. Our approach addresses the efficiency-heterogeneity trade-off through two key designs. At the prompt level, we propose a novel "Group-and-Distill" strategy. By leveraging the in-context learning capabilities of LLMs, we aggregate agents with similar static profiles into shared contexts while maintaining their unique dynamic memories, significantly reducing token redundancy without sacrificing individual distinctiveness. At the system level, we implement a request scheduler that optimizes I/O concurrency, effectively neutralizing the latency of massive API calls.

We evaluate the efficiency and faithfulness of OpenCity in simulating the urban activities of six global cities using the widely adopted Generative Agent workflow [16]. Our experiments show OpenCity achieves an average 635x acceleration in simulations with 10,000 LLM agents on commodity hardware. Besides, the number of requests and consumed tokens are reduced by 73.7% and 45.5%, respectively. OpenCity also shows strong scalability, with the simulation time per agent reducing from 36.25 to 0.06 seconds as the simulation size increases from 1 to 10,000 agents, demonstrating that larger simulations allow for more efficient LLM request scheduling and prompt distillation. Importantly, OpenCity maintains high faithfulness of the simulated behaviour. The Jensen–Shannon divergence and top-1 hit rates of our method are comparable to the standard prompting technique of batch prompting [2], and substantially surpass straightforward reusing strategy [3]. Besides, the top-1 hit rate can reaches up to 96% when using powerful LLMs like GPT-4o, demonstrating that OpenCity can preserve high individuality of the agent.

The substantial simulation acceleration allows us to benchmark the *experienced segregation* of global cities using LLM agents for the first time. Our experiments reveal that OpenCity not only reproduces physical mobility patterns but also captures socioeconomic segregation significantly better than classic rule-based models (e.g., Social-EPR [14]). Furthermore, our scalability analysis quantitatively validates that emergent social phenomena become increasingly distinguishable as simulation scale and duration grow, underscoring the necessity of large-scale agent populations. Finally, we demonstrate the framework's potential as a "policy sandbox" by conducting counterfactual experiments to evaluate urban equity interventions.

The main contributions of this work are summarized as follows:

- We propose OpenCity, which integrates a "Group-and-Distill" strategy with a high-throughput scheduler. This achieves a 600× speedup and 45% token reduction while preserving the agent heterogeneity essential for social simulation, enabling the simulation of 10,000 agents on commodity hardware.
- We conduct comprehensive experiments across six global cities. Results demonstrate that LLM agents, accelerated by our scalable framework, not only maintain high micro-level fidelity but also significantly outperform classic rule-based models (e.g., Social-EPR) in capturing emergent socioeconomic segregation.
- We quantitatively validate that simulation fidelity improves convergently with scale, proving the necessity of large-scale agent populations. Furthermore, we demonstrate the framework's utility as a "policy sandbox" for counterfactual urban analysis.

2 Related Works

2.1 LLM Agents

With the widespread use of large language models (LLMs) in various applications, the limitations of LLMs, e.g., unstable reasoning abilities, limited memory capacity, and lack of specialized expertise, have been exposed to the public. As one of the potential solution, LLM agents are proposed to overcome these limitations and promote the practical application of LLMs. AutoGPT [7] as one of the most popular LLM autonomous agent explore the potential of applying LLM to enable the autonomous planning and task-solving. After that, LLM agents [24, 27] have made significant progress in two directions: task-oriented agents and simulation agents. Following the first direction, researchers aim to improve LLM agent's ability to solve complex tasks [11, 21]. For example, lots of programming agents, such as ChatDev [17], SWEAgent [30], and MetaGPT [8], are designed to solve the complex programming tasks. As for the second direction, generative agents [16] have demonstrated the potential of large models in simulating human behavior, which has been further validated in subsequent research. S3 [5] explores the potential of using LLM agents to simulate the social network. CoPB [22] defines a agentic workflow to simulate the mobility behaviors. RecAgent [25] simulate the user behavior in the recommendation system. While these works demonstrate the potential of LLM agents, the large scale efficient simulation of generative agents becomes the critical bottleneck of further applications. This work aims to fill this critical gap, proposing a platform that makes

large-scale LLM agent simulation not just theoretically possible, but practically achievable.

2.2 LLM Deployment Optimization

To support the efficient inference of LLMs and LLM agents, enormous works and systems [13] are designed to optimize the inference efficiency of LLMs and further accelerate their practical applications. For example, Flash-attention [4] is an IO-aware exact attention algorithm which uses tiling to reduce the number of memory reads/writes within GPU. AWQ [12] is an activation-aware weight quantization to compress and accelerate the LLM inference. vLLM [10] proposes pagedAttention mechanism to enable highly efficient KV cache scheduler during the inference and becomes the most population open source LLM inference engine. SGLang [32] provides a flexible frontend language to enable the efficient autonomous optimization of LLM inference. Synergy-of-thought [20] proposes to exploit the synergy between larger and smaller language models for efficient reasoning. While these systems are designed to process the general LLM inference, specific characteristics of generative agents especially urban generative agents are ignored which can be employed to further accelerate the inference and simulation. In this paper, we explore the potential of this direction and design the OpenCity platform.

3 Problem Formulation

3.1 LLM Agents for Urban Activities

We aim to simulate urban dynamics by modeling the behavior of a population of N LLM agents within an urban environment E. The state of an agent i at time t is denoted by $S_i(t)$, which we decompose into two conceptually distinct components:

$$S_i(t) = \{s_i, m_i(t)\} \tag{1}$$

Here, s_i represents the **static properties** of the agent, such as demographics (e.g., age, occupation), which remain constant throughout the simulation. In contrast, $m_i(t)$ represents the **dynamic properties**, including the agent's memory stream and its perception of the environment, which evolve over time.

The simulation proceeds in discrete time steps, where an agent's behavior is governed by a state update function f. This function is not a simple calculation but an entire cognitive process, primarily implemented via prompting a large language model:

$$S_i(t+1) = f(S_i(t), E) \tag{2}$$

Conceptually, the function f includes the agent's decision-making cycle, which involves several steps: **Perception**, where the agent processes information about its current surroundings from E; **Reflection**, where it retrieves relevant past experiences from its memory $m_i(t)$; and **Action Planning**, where it synthesizes this information to decide on its next action, thereby updating its dynamic state from $m_i(t)$ to $m_i(t+1)$. The sequence of an agent's physical locations over time, $T_i = \{L_i(0), L_i(1), \ldots, L_i(t_s)\}$, forms its trajectory. Along with individual mobility, we also examine the aggregated mobility features $A = \{S_i(t)\}_{i,t}$, such as Original-Destination (OD) matrix and income segregation index, which reflects the urban dynamics involving states of all agents.

3.2 The Scalability Bottlenecks Analysis

To simulate urban dynamics, the state update function (Equation 2) must be applied to all N agents at each time step. When N grows to thousands (a scale necessary for meaningful urban analysis), this process faces two critical, interconnected bottlenecks that render naive implementations intractable. The first is **the prohibitive cost of ensuring agent heterogeneity at scale**, which prevents simple response reuse. The other is **the severe I/O latency inherent in remote model invocations**. To empirically validate these bottlenecks, we conducted a preliminary analysis, with the results shown in Appendix Figure A1.

The challenge of ensuring heterogeneity is an application-level bottleneck. As established, in the urban environment an agent's state $S_i(t)$ includes a dynamic memory component $m_i(t)$ that evolves with every action. This continuous evolution requires a distinct LLM request to update each agent's state, ensuring that its future actions are grounded in its unique history. A seemingly obvious optimization is to reuse a single LLM response for multiple "similar" agents [3]. However, this strategy is fundamentally incompatible with realistic social simulation due to the path-dependent nature of agent experience. This necessitates a more nuanced approach. To investigate the potential for such an approach, we analyzed the semantic and syntactic properties of agent prompts. The t-SNE visualization in Figure A1(a) clearly shows that agents' static profiles form distinct semantic clusters. Figure A1(b)'s cumulative distribution plots confirm that agent profiles are semantically coherent (Embedding Distance, solid lines, skewed left) yet syntactically diverse (Edit Distance, dashed lines, skewed right). This finding confirms that while agents are too unique for simple reuse, their underlying similarities present a clear opportunity for a group-based optimization strategy.

The second bottleneck is technical: the severe I/O latency of remote model invocations, which makes the simulation an overwhelmingly **I/O-bound** process. As profiled in Figure A1(c), over **99%** of a naive simulation's runtime is consumed by network I/O (waiting for remote server responses), while local CPU computation accounts for less than 1%. Large-scale simulations require many such requests, and these waiting times greatly reduce efficiency. The figure also starkly illustrates the potential of concurrency: a simple parallel implementation with 50 workers reduces the total time from 442.6 seconds to just 19.06 seconds. It confirms that system resources also remain underutilized. Thus, effective scheduling of LLM requests is crucial for improving both resource utilization and overall simulation efficiency. Furthermore, since inference cost is proportional to token count, a comprehensive solution must address both the high latency of individual calls and the cumulative token consumption of the entire agent population.

Based on these observations, this work introduces a novel prompt distillation method and an efficient LLM request scheduler to significantly improve the efficiency of large-scale LLM agent simulations.

4 OpenCity Framework

To address the scalability challenges of large-scale agent simulation, we developed OpenCity, a framework that improves efficiency on two complementary levels. At the prompt level, we propose a novel "group-and-distill" Optimizer to reduce token consumption and

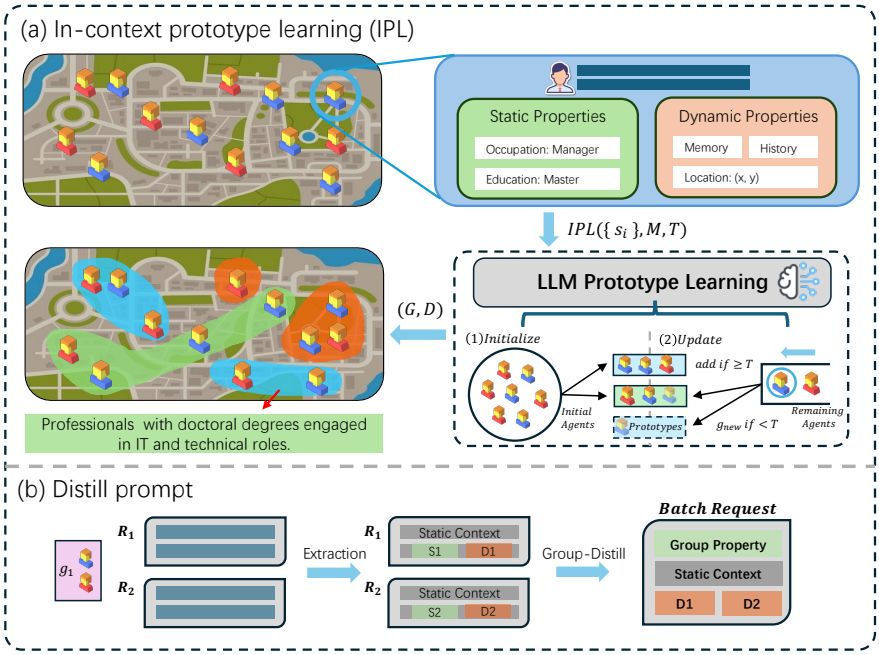

Figure 1: Overview of Group-and-Distill Meta-Prompt Optimizer.

request counts while preserving agent individuality. At the system level, we introduce a high-throughput LLM Request Scheduler to manage network operations and mitigate the high latency of API calls. Together, these components enable efficient simulation at an unprecedented scale while maintaining high behavioral fidelity. We describe each in turn.

4.1 Group-and-Distill Meta-Prompt Optimizer

The sheer number of requests and tokens in large-scale simulation remains a significant cost and performance bottleneck. A naive solution like reusing a single LLM response for multiple agents is untenable, as it compromises the agents' individuality and behavioral fidelity, a core tenet of ABM. Our key insight is to separate an agent's prompt into two parts: a **shared context** derived from its immutable *static properties* (s_i), and a **unique payload** containing its evolving *dynamic properties* ($m_i(t)$) as illustrated in Figure 1. By grouping agents with similar static properties, we can send the shared context only once for the entire group, drastically reducing token count, while the unique payload ensures each agent acts based on its own experience. This is achieved through a two-stage process detailed below.

Stage 1: Agent Grouping via In-context Prototype Learning (IPL). The first stage is to cluster the population of N agents into meaningful groups based on their static profiles s_i. Traditional clustering algorithms (e.g., k-means) are not enough for this task, as they require converting rich, textual profiles into numerical feature vectors, a process that often loses critical semantic nuance (e.g. different group have different lifestyles). We instead leverage the LLM's own powerful in-context semantic understanding to perform the clustering in a zero-shot manner. In this way, we use the semantic prototypes rather than the abstract centroids to ensure that the grouping logic remains aligned with the social variables being modeled. The process is formalized in Algorithm 1 in Appendix. To maintain system scalability, IPL is implemented as a one-time offline initialization step. We optimize the process by using batched similarity queries, where the LLM evaluates an agent's profile against multiple candidate prototypes in a single prompt, rather than through individual pairwise comparisons.

Hyperparameter Discussion. The IPL algorithm has two key hyperparameters. M is a small integer (e.g., 20-50) used to seed the initial set of prototypes. Its exact value has minimal impact on the final clustering for a large population. T is the similarity threshold (e.g., 0.8), which controls the granularity of the agent groups. A higher T results in more, smaller, and more homogeneous groups, while a lower T results in fewer, larger, and more diverse groups. We selected $T = 0.8$ based on empirical validation to achieve a good balance between intra-group similarity and the total number of groups.

Stage 2: Distilled Prompt Generation and Execution. With agents clustered into groups, we can now "distill" the prompts at runtime. A conventional, unoptimized prompt for a single agent would contain its entire verbose static profile, followed by its dynamic state. For example: *"Your full profile is: You are Isabella, a 28-year-old software engineer... Your current memory is: ..."*. This leads to significant redundancy when repeated for thousands of agents.

Our distilled prompt, in contrast, is structured as a single batch request for an entire group. It consists of two parts. The prompt begins with a **Shared Context** derived from the group description d_g generated by the IPL algorithm, such as: *"You are all part of a group of: 'Young, high-income tech professionals living downtown who prefer quiet recreational activities.' Based on this shared identity and your unique individual states below, decide on your plan..."*. This is

followed by a series of **Individual Payloads**, where each payload contains only the dynamic properties $m_i(t)$ for a specific agent in the group (e.g., *"Person 1 (Isabella): Memory: ..., Location: ..."*). This aggregated request is sent to the LLM, which processes the batch and returns a corresponding set of individual decisions. This method drastically reduces the number of API calls (from one per agent to one per group) and total token count by eliminating the redundant transmission of static profile information.

4.2 LLM Request Scheduler

The other primary performance limitation is the cumulative time spent waiting for network responses from thousands of individual API calls. Our LLM Request Scheduler is designed specifically to mitigate this bottleneck through three complementary techniques.

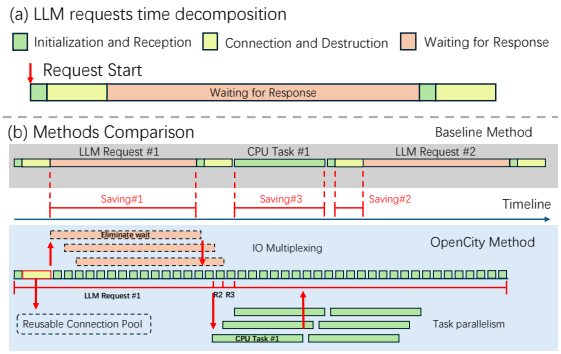

Figure 2: The functionality of the LLM Request Scheduler.

Asynchronous I/O with Multiplexing: Our scheduler is built on a non-blocking event loop using efficient I/O notification mechanisms (e.g., epoll on Linux [1]). Agent requests are dispatched asynchronously, allowing the main simulation thread to continue processing other agents and I/O events without blocking. Network responses are handled only when ready, enabling a single thread to manage thousands of concurrent LLM requests and effectively hide network latency (Time saving #1 in Figure 2(b)).

Reusable Connection Pooling: To avoid the overhead of repeatedly establishing TCP connections, we maintain a pool of persistent connections to the LLM service. Agents borrow and return connections as needed, amortizing connection setup costs and reducing per-request latency (Time saving #2 in Figure 2(b)).

Decoupled CPU-Bound Task Execution: LLM calls are I/O-bound, but agents also perform CPU-bound local computations. To prevent these tasks from blocking the event loop, all non-trivial CPU-bound operations are offloaded to a worker thread pool. This design keeps the I/O loop responsive and ensures efficient parallel execution (Time saving #3 in Figure 2(b)).

Collectively, these techniques transform the simulation from a slow, sequential process into a highly parallel and efficient system capable of handling massive-scale agent-environment interactions.

5 Experiments

5.1 Dataset and Setup

Datasets. We conduct our experiments on urban mobility data from six major cities: Beijing, New York, San Francisco, London, Paris, and Sydney. The data sources are diverse to ensure the generality of our findings. The Beijing's data comes from a related work [22], which collected from social network platform. The New York and San Francisco data consist of aggregated population flows from Safegraph, which is a wide-used dataset for segregation research. Data for the remaining three cities are based on user check-ins from Foursquare. All datasets underwent a standardized preprocessing pipeline, including trajectory filtering, home location extraction, and user profile sampling based on census data where necessary. Further details on each dataset and the preprocessing steps are provided in Appendix A.

LLM Agent Configuration. The agents in OpenCity are implemented based on the widely adopted "Generative Agent" architecture [16], which encompasses a perception-planning-reflection cognitive cycle. At each step, an agent perceives its environment, makes a plan, and acts upon it, recording the experience into a memory stream for future reflection. Unless otherwise specified, the underlying large language model for all agents in our experiments is **OpenAI's GPT-4o-mini**, accessed via their official API. For the faithfulness analysis presented in Section 5.2.2, we also report results using the more powerful **GPT-4o** model to evaluate the impact of model capability.

Baselines for Comparison. We evaluate OpenCity against two distinct types of baselines, each serving a different purpose:

- **Performance Baseline (Unoptimized LLM Agent):** To quantify the acceleration achieved by OpenCity, our primary performance baseline is an *unoptimized LLM agent implementation*. This baseline follows the standard generative agent framework but makes **serial, blocking API calls** for each agent's decision step. It does not include any of OpenCity's system-level scheduling or prompt-level optimizations and thus represents the naive approach to implementing a large-scale simulation.
- **Urban Dynamics Baseline:** To validate the fidelity of our simulation's outcomes, we compare it against three baselines: (i) the Explore and Preferential-Return (EPR) model [23], a classic rule-based model, which determines an agent's next location based on a probabilistic choice between exploring new locations and returning to previously visited ones. For our experiments, we use the standard parameter settings: exploration rate $\rho = 0.6$, trade-off parameter $\gamma = 0.21$, and waiting time distribution parameters $\tau = 17, \beta = 0.8$. (iii) the Social-EPR model [14], which is incorporates a "social exploration" parameter (σ_s) in EPR to better simulate experienced income segregation. (ii) the CoPB model [22], a hybrid method that uses LLMs for activity planning and reverts to Gravity Models for location selection.

5.2 OpenCity Platform Evaluation

We conduct a two-part evaluation of the OpenCity. First, we measure its performance in terms of simulation speed and scalability. Second, we assess the fidelity of the simulated agent behaviors to ensure our optimizations do not compromise the quality of the results.

5.2.1 Performance Evaluation: Scalability and Efficiency. This section quantifies the acceleration and resource savings provided by

Table 1: Faithfulness experiment results

Model and Cities		Inherent		Batch prompting		Archetype prompting		Ours	
		JSD	*T*1	*JSD*	*T*1	*JSD*	*T*1	*JSD*	*T*1
4o-mini	Beijing	0.04 ± 0.02	90%	0.11 ± 0.05	76%	0.89 ± 0.04	8%	0.13 ± 0.02	74%
	New York	0.02 ± 0.01	92%	0.07 ± 0.03	81%	0.84 ± 0.11	13%	0.06 ± 0.04	86%
	San Francisco	0.03 ± 0.02	88%	0.09 ± 0.04	77%	0.91 ± 0.03	11%	0.10 ± 0.03	85%
	London	0.06 ± 0.04	89%	0.12 ± 0.07	79%	0.86 ± 0.06	9%	0.12 ± 0.04	78%
	Paris	0.05 ± 0.02	86%	0.17 ± 0.11	69%	0.94 ± 0.03	4%	0.14 ± 0.04	71%
	Sydney	0.04 ± 0.03	85%	0.08 ± 0.03	75%	0.88 ± 0.05	5%	0.07 ± 0.04	75%
GPT-4o	New York	0.003 ± 0.002	98%	0.012 ± 0.007	94%	0.89 ± 0.09	10%	0.009 ± 0.004	97%
	Paris	0.004 ± 0.002	99%	0.021 ± 0.009	93%	0.91 ± 0.04	7%	0.010 ± 0.006	96%

Table 2: Acceleration experiment results

Cities	Time	*Speedup*	*Rr*	*Tr*
Beijing	0.07s	521.7	73.2%	38.7%
New York	0.06s	624.7	67.3%	37.6%
San Francisco	0.07s	588.6	80.3%	51.3%
London	0.04s	792.5	74.6%	49.9%
Paris	0.06s	640.0	76.3%	48.6%
Sydney	0.05s	644.0	70.7%	46.6%
Average	**0.058s**	**635.3**	**73.7%**	**45.5%**

OpenCity. We compare its performance against the **unoptimized LLM agent baseline** defined in our setup, which makes serial, blocking API calls.

Overall Acceleration. The performance was evaluated in six major cities, each with 10,000 agents. The test environment was a cloud server with an Intel Xeon Platinum 8378C CPU and 256 GB RAM. The results are presented in Table 2. We measure four key metrics: simulation time per agent (Time), the overall speedup factor (Speedup), the reduction rate in LLM API requests (Rr), and the reduction rate in total tokens consumed (Tr). The substantial average speedup of **635.3x** stems from two synergistic dimensions: system-level parallelization and algorithmic compression. Specifically, the Request Scheduler manages high-concurrency asynchronous I/O to mitigate network latency, while the IPL module achieves a fundamental reduction in the computational workload on requests and tokens. The results demonstrate that IPL markedly reduces API calls and token consumption by an average of **73.7%** and **45.5%**, respectively. Together with these two layers, OpenCity reduces the average runtime per agent to just 0.058 seconds, enabling city-scale agent simulation that was previously computationally prohibitive.

Scalability Analysis. To assess how OpenCity's performance scales with the number of agents, we conducted simulations with agent populations ranging from 10 to 10,000. The results are shown in Figure 3(a). The platform demonstrates strong scalability, with the time-per-agent decreasing significantly as the population grows. This is because larger simulations provide greater opportunities for both of our optimization strategies: the LLM request scheduler can more effectively hide network latency with a larger pool of concurrent requests, and the "group-and-distill" optimizer can form more and larger groups, leading to a higher token-saving efficiency.

5.2.2 Fidelity Evaluation: Preserving Agent Individuality. A fast simulation is only useful if its results are believable. This section evaluates whether our "Group-and-Distill" optimizer preserves the nuanced, individualistic behavior of the agents.

Experimental Design. The testbed for this evaluation is location choice generation. We randomly selected 100 agents and, for each agent, prompted for a location choice 100 times using the same context. We compared four methods: (1) **Raw Prompting**, which is the unoptimized, one-agent-one-prompt method; (2) **Batch Prompting** [2]; (3) **Archetype Prompting** [3], which represents the naive reuse strategy; and (4) **Ours (OpenCity)**.

Metrics. We use two metrics to evaluate fidelity:

- **Jensen-Shannon Divergence (JSD):** This measures the difference between the distribution of an agent's 100 choices using an optimized method versus the "ground truth" distribution from Raw Prompting. A lower JSD indicates higher fidelity.
- **Top-1 Hit Rate (T1):** This measures the consistency of an agent's single most frequent choice, reflecting the stability of its core preferences.

Results and Analysis. The results are shown in Table 1. Here, the 'Inherent' row denotes the inter-run consistency of the unoptimized Raw Prompting, serving as a reference for the upper bound of behavioral stability. As expected, the Archetype Prompting method performs poorly, with high JSD, demonstrating its inability to capture agent individuality. In contrast, our method achieves a JSD and T1 rate comparable to the standard Batch Prompting technique, confirming its ability to maintain high behavioral fidelity. To investigate the impact of the underlying model's capability, we conducted a focused experiment on two cities using the more powerful GPT-4o model. This test was limited to two cities due to the significantly higher computational cost and API expense of GPT-4o. The findings indicate that with a more capable model, our method's output closely approximates that of Raw Prompting, achieving a T1 rate of up to 97%.

In summary, the evaluation demonstrates that OpenCity not only enhances the efficiency of large-scale simulations by orders of magnitude but also does so while preserving the distinctive characteristics of the agents, making it a robust and reliable platform for large-scale social simulation.

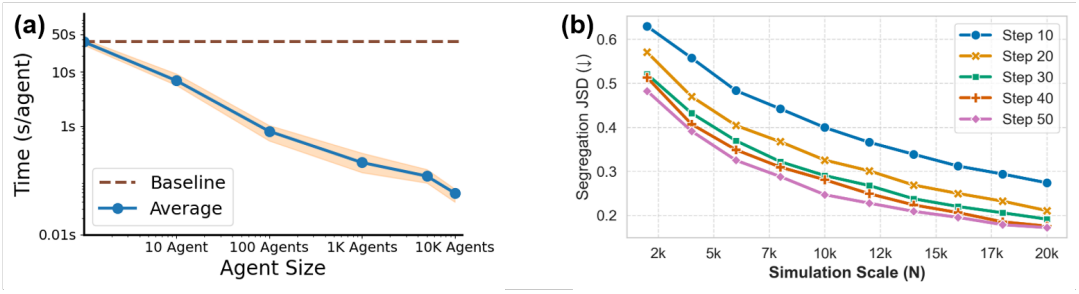

Figure 3: Scalability experiments. (a) Runtime scalability, and (b) simulation fidelity under increasing scale.

5.3 Reproducing Urban Dynamics

The efficiency of OpenCity enables us, for the first time, to benchmark the ability of large-scale LLM agents to reproduce real-world urban dynamics. We use a comprehensive, multi-level set of metrics to evaluate the simulation fidelity. At the **individual level**, we measure the Jensen-Shannon Divergence (JSD) of the Radius of Gyration (r_g) distributions [6]. At the **group level**, we assess the structural similarity of city-wide flows using the Common Part of Commuters (CPC) of the Origin-Destination (OD) matrix [9]. In the **social domain**, we evaluate the emergence of socioeconomic patterns using both the JSD and Correlation (Coor) of the experienced income segregation index. [14]. Further details on these metrics are available in Appendix B.

We analyze the performance of our LLM-based Generative Agent against the EPR/Social-EPR and CoPB Agent in six major cities, each simulated with 1,000 agents. The results are presented in Table 3. Note that the $r_g(JSD)$ metric, which is calculated from individual trajectories, is not applicable (N/A) for New York and San Francisco, as their SafeGraph datasets provide aggregated population flows rather than individual user paths.

The results indicate that the LLM-powered Generative Agent is highly competitive with, and in several aspects superior to, the well-established EPR/Social-EPR model and LLM-based CoPB Agents.

- On **individual mobility** (r_g), the OpenCity-GA generally capture the spatial extent of individual human movement, achieve the competitive JSD against EPR/Social EPR Model. While it performs slightly worse than CoPB in some cities, this is because CoPB's hybrid architecture is explicitly designed to optimize for physical distance distributions, whereas our GA is driven by semantic reasoning.
- On **aggregate flows** (OD), the GA achieve superior performance in five out of six cities. This indicates that the semantic advantage of LLMs becomes more obvious as individual behaviors aggregate into macro-level flows.
- On **social dynamics** ($Segregation$), the Generative Agent generally achieves its most significant advantage in reproducing the income segregation. While Social-EPR is carefully designed to reproduce segregation by introducing social exploration factors, the GA's accurate segregation patterns emerge naturally from high-fidelity, autonomous decision-making. This validates the necessity of cognitive micro-foundations in modeling complex socioeconomic drivers of location choice.

The Necessity of Scale. The scientific utility is further demonstrated by the scalability analysis in New York as illustrated in Figure 3(b). The results show a clear trend: as the simulation scale (N) increases from 2,000 to 20,000 agents, the JSD monotonically decreases. This quantitatively validates that emergent urban phenomena are population-level properties that become distinguishable only at a certain magnitude. By enabling simulations at the scale of $N \geq 10^4$, OpenCity provides the necessary infrastructure to move into more meaningful city-scale scientific discovery.

6 Case Study

OpenCity supports two capabilities that are difficult for conventional models: large-scale counterfactual experimentation and micro-level analysis through direct interrogation of individual agents. We demonstrate these capabilities in a case study on experienced urban segregation. Experienced urban segregation arises from a complex interplay between where people live (residential segregation) and where they go (mobility patterns) [14]. A key policy question is to disentangle the impact of these two factors. We use OpenCity to conduct a counterfactual experiment in New York and San Francisco to investigate this.

Experimental Design. We simulate two scenarios. The **'Original'** scenario initializes agent home locations based on real-world census data, reflecting the existing residential segregation. The **'Even'** counterfactual scenario programmatically eliminates residential segregation by evenly distributing agents from all income levels across the city's neighborhoods, while keeping their behavioral models unchanged.

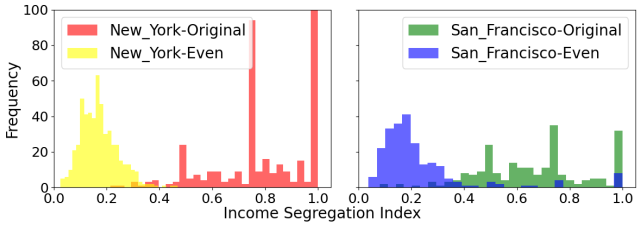

Figure 4: The distribution of income segregation index.

Results and Interpretation. The results, shown in Figure 4, reveal a dramatic shift. In New York City, eliminating residential segregation causes the mean experienced segregation index to decrease from 0.845 to 0.172. Similarly, in San Francisco, the index drops from 0.665 to 0.232. This result indicates that, under the assumption of stable behavioral preferences, the structural factor of

Table 3: Benchmarking Urban Dynamics across Six Cities.

City	Beijing				New York				San Francisco			
Metric	r_g	OD	Segregation		r_g	OD	Segregation		r_g	OD	Segregation	
Method	*JSD* ↓	*CPC* ↑	*JSD* ↓	*Coor* ↑	*JSD* ↓	*CPC* ↑	*JSD* ↓	*Coor* ↑	*JSD* ↓	*CPC* ↑	*JSD* ↓	*Coor* ↑
EPR	0.427	0.299	0.166	0.317	-	0.088	0.534	0.0511	-	0.181	0.330	0.254
Social-EPR	0.339	0.291	0.171	0.120	-	0.092	0.560	0.1025	-	0.149	0.309	**0.285**
CoPB	**0.069**	0.298	0.141	0.098	-	0.066	0.588	0.1236	-	0.149	0.319	0.203
OpenCity-GA	0.221	**0.327**	**0.126**	**0.572**	-	**0.157**	**0.421**	**0.1352**	-	**0.242**	**0.299**	0.232

City	Paris				London				Sydney			
Metric	r_g	OD	Segregation		r_g	OD	Segregation		r_g	OD	Segregation	
Method	*JSD* ↓	*CPC* ↑	*JSD* ↓	*Coor* ↑	*JSD* ↓	*CPC* ↑	*JSD* ↓	*Coor* ↑	*JSD* ↓	*CPC* ↑	*JSD* ↓	*Coor* ↑
EPR	0.528	0.159	0.159	0.048	0.491	0.168	0.154	**0.270**	0.487	0.161	0.257	0.314
Social-EPR	**0.017**	0.156	0.148	0.087	0.481	**0.176**	0.172	0.149	0.486	0.155	0.278	0.341
CoPB	0.459	0.150	0.158	0.041	0.456	0.159	**0.128**	0.265	**0.092**	0.143	**0.211**	0.126
OpenCity-GA	0.363	**0.194**	**0.102**	**0.419**	**0.362**	0.121	0.193	0.213	0.355	**0.191**	0.221	**0.401**

residential location remains the dominant driver of experienced segregation within our simulation framework. This demonstrates the platform's utility for policymakers to test hypotheses and estimate the potential impact of interventions, such as policies promoting mixed-income housing.

Interrogating Agent Motivations. A unique capability of LLM agents is that they are **interrogable**. Beyond aggregate statistics, OpenCity enables researchers to query individual agents in natural language after simulation to uncover the reasoning behind their behaviors. As shown in Figure A3 (Appendix), agents can narrate their daily activities, justify decisions (e.g., "enjoying a leisurely meal"), and recall social interactions. While these responses represent the LLM's generative reasoning rather than verified biological intent, they provide valuable interpretability for agent-based simulations, allowing researchers to examine the internal consistency of the simulated social dynamics. This capability transforms agents from mere data points into virtual subjects that can be "interviewed." It allows researchers to move beyond quantitative metrics and collect rich, qualitative data on agents' perceived experiences, social circles, and decision-making logic.

7 Conclusion

In this work, we introduced OpenCity, a scalable framework designed to tackle the computational and communication challenges of deploying large-scale LLM-based urban agents in city simulations. By integrating a novel "group-and-distill" prompt optimization strategy and an LLM request scheduler, we achieved a 600× increase in simulation efficiency, with major reductions in both LLM requests and token usage. We evaluated OpenCity through experiments in six global cities. The results showed that the platform can simulate the daily activities of 10,000 agents at an hourly level, setting a new benchmark for generative agent performance in urban environments. OpenCity's ability to align simulated behaviors with real-world data highlights its promise for practical, city-scale applications. Beyond large-scale counterfactual analyses, the platform uniquely enables researchers to interact directly with individual agents in natural language to explore their motivations—combining quantitative scale with qualitative insight. As a robust and extensible foundation, we believe OpenCity will empower the research community to study complex societal dynamics at a scale and depth that were previously out of reach.

Limitations and Ethical Considerations

Despite the efficiency gains of OpenCity, several limitations remain. First, while the framework supports city-scale simulations with reduced computational cost, scaling to fully million-level populations remains challenging under current LLM inference budgets. Moreover, since inherent socioeconomic and cultural biases in LLMs may influence simulated dynamics and our validation benchmarks rely on existing mobility models, simulation results should be interpreted as heuristic insights into complex urban systems rather than exact real-world predictions. Regarding data ethics, all real-world datasets utilized in our experiments were rigorously anonymized and aggregated to ensure that no individual-level sensitive information could be reconstructed.

Contribution Statement

This project was a cross-disciplinary collaboration among researchers from the Center for Urban Science and Computation at Tsinghua University, the Information Hub at HKUST (GZ), and the Department of Sociology at the University of Chicago. Q. Zeng and Y. Yan designed the simulation framework and conducted the majority of the experiments. Z. Zhen, J. Yuan, J. Feng, and J. Zhang carried out baseline experiments and assisted with data visualization. J. Evans offered critical conceptual insights from sociology and inform experiment design. F. Xu and Y. Li conceived the overall research idea and initiated this study. All authors contributed to manuscript preparation and revision, and approved the final version.

GenAI Disclosure

Large Language Models (LLMs), specifically GPT-4o and GPT-4o-mini, serve as the core cognitive engine within the OpenCity framework to simulate agent reasoning, planning, and social interactions. And they were utilized in the In-context Prototype Learning (IPL) phase to perform zero-shot semantic clustering of agent profiles. The simulation platform generates synthetic mobility trajectories and social encounter data based on the cognitive outputs of LLM agents. LLM tools (Gemini) were employed during the manuscript preparation process for grammatical polishing and technical phrasing refinement. All scientific claims, experimental interpretations, and the architectural design of the OpenCity system remain the original work of the authors.

References

[1] Francesc Bruguera i Moriscot. 2019. Benchmarking input/output multiplexing facilities of the Linux kernel. (2019).

[2] Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch Prompting: Efficient Inference with Large Language Model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*. 792–810.

[3] Ayush Chopra, Shashank Kumar, Nurullah Giray Kuru, Ramesh Raskar, and Arnau Quera-Bofarull. 2025. On the Limits of Agency in Agent-based Models. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. 500–509.

[4] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.

[5] Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. S^3: Social-network Simulation System with Large Language Model-Empowered Agents. *arXiv preprint arXiv:2307.14984* (2023).

[6] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *nature* 453, 7196 (2008), 779–782.

[7] Significant Gravitas. 2023. AutoGPT. https://github.com/Significant-Gravitas/AutoGPT. Accessed: 2024-09-01.

[8] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*.

[9] Shan Jiang, Yingxiang Yang, Siddharth Gupta, Daniele Veneziano, Shounak Athavale, and Marta C González. 2016. The TimeGeo modeling framework for urban mobility without travel surveys. *Proceedings of the National Academy of Sciences* 113, 37 (2016), E5370–E5378.

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.

[11] Yu Li, Lehui Li, Zhihao Wu, Qingmin Liao, Jianye Hao, Kun Shao, Fengli Xu, and Yong Li. 2025. AgentSwift: Efficient LLM Agent Design via Value-guided Hierarchical Search. *arXiv preprint arXiv:2506.06017* (2025).

[12] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems* 6 (2024), 87–100.

[13] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2025. Towards efficient generative large language model serving: A survey from algorithms to systems. *Comput. Surveys* 58, 1 (2025), 1–37.

[14] Esteban Moro, Dan Calacci, Xiaowen Dong, and Alex Pentland. 2021. Mobility patterns are associated with experienced income segregation in large US cities. *Nature communications* 12, 1 (2021), 4633.

[15] Hamed Nilforoshan, Wenli Looi, Emma Pierson, Blanca Villanueva, Nic Fishman, Yiling Chen, John Sholar, Beth Redbird, David Grusky, and Jure Leskovec. 2023. Human mobility networks reveal increased segregation in large cities. *Nature* 624, 7992 (2023), 586–592.

[16] 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*. 1–22.

[17] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924* 6 (2023).

[18] Thomas C Schelling. 1969. Models of segregation. *The American economic review* 59, 2 (1969), 488–493.

[19] Thomas C Schelling. 1971. Dynamic models of segregation. *Journal of mathematical sociology* 1, 2 (1971), 143–186.

[20] Yu Shang, Yu Li, Fengli Xu, and Yong Li. 2024. DefInt: A Default-interventionist Framework for Efficient Reasoning with Hybrid Large Language Models. *arXiv preprint arXiv:2402.02563* (2024).

[21] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. 2024. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153* (2024).

[22] Chenyang Shao, Fengli Xu, Bingbing Fan, Jingtao Ding, Yuan Yuan, Meng Wang, and Yong Li. 2024. Beyond imitation: Generating human mobility from context-aware reasoning with large language models. *arXiv preprint arXiv:2402.09836* (2024).

[23] Chaoming Song, Tal Koren, Pu Wang, and Albert-László Barabási. 2010. Modelling the scaling properties of human mobility. *Nature physics* 6, 10 (2010), 818–823.

[24] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[25] Lei Wang, Jingsen Zhang, Xu Chen, Yankai Lin, Ruihua Song, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Recagent: A novel simulation paradigm for recommender systems. *arXiv preprint arXiv:2306.02552* (2023).

[26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[27] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences* 68, 2 (2025), 121101.

[28] Fengli Xu, Qi Wang, Esteban Moro, Lin Chen, Arianna Salazar Miranda, Marta C González, Michele Tizzoni, Chaoming Song, Carlo Ratti, Luis Bettencourt, et al. 2025. Using human mobility data to quantify experienced urban inequalities. *Nature Human Behaviour* (2025), 1–11.

[29] Fengli Xu, Jun Zhang, Chen Gao, Jie Feng, and Yong Li. 2023. Urban generative intelligence (ugi): A foundational platform for agents in embodied city environment. *arXiv preprint arXiv:2312.11813* (2023).

[30] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems* 37 (2024), 50528–50652.

[31] Qingbin Zeng, Qinglong Yang, Shunan Dong, Heming Du, Liang Zheng, Fengli Xu, and Yong Li. 2024. Perceive, Reflect, and Plan: Designing LLM Agent for Goal-Directed City Navigation without Instructions. *arXiv preprint arXiv:2408.04168* (2024).

[32] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2023. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104* (2023).

A Urban Mobility Dataset

As shown in Table A1, we collect urban mobility data of 6 major cities around the world. The data sources vary. In Beijing, the data is from a related work [22], which gathered through a social network platform and tracking users' mobility trajectories. Additionally, users' profiles, such as income level, gender, occupation, education level and age, are collected through digital surveys. In New York and San Francisco, the data comes from Safegraph, which provides aggregated population flow among Points of Interest(POIs) and Census Block Groups(CBGs). The other three cities—London, Paris and Sydney—use data are from Foursquare. Foursquare data consist of thousands of check-ins data of users and the corresponding venue position.

To make better use of the datasets, we apply several preprocessing methods. We firstly arrange the trajectory points in time sequence, and divide the trajectory into units of one day. Then we filter out trajectories with fewer than 4 points in a day, as they do not fully capture users' mobility patterns. For home extraction, we identify the most frequently visited location of the useres the home. Since only the Tencent dataset includes user profiles, we make profile sampling for users of each city based on local census data for the other two datasets. In the end, our dataset is optimized for easy use in urban mobility simulations.

B Urban dynamic metrics

We use comprehensive metrics in three-levels to evaluate the simulation performance, from individual-level to group level, and also from physical domain to social domain. These metrics allows us to gain a full understanding of mobility patterns and their implications, and can also help us evaluate the performance of the simulation by analysing the generated trajectory.

At the individual level, we calculate radius of gyration r_g [6] for each user, which is a measure of the spatial extent of their movements. The radius of gyration is defined as follows:

$$r_g^\alpha = \sqrt{\frac{1}{N^\alpha}\sum_{i=1}^{N^\alpha}(\vec{r}_i^\alpha - \vec{r}_{cm}^\alpha)^2} \tag{3}$$

where $\vec{r}_i^\alpha$ represents the $i = 1, 2, ..., N$ positions recorded by user α, and $r_{cm}^\alpha = 1/N^\alpha \sum_{i=1}^{N^\alpha}(\vec{r}_i^\alpha)$ is the center of mass of the trajectory. The radius of gyration provides an indication of the size of a user's activity range.

To analyze movement patterns and other aggregated features, we define block areas as spatial units within the city. For cities with Safegraph data, we use existing Census Block Group (CBG) areas. For other cities, we divide the map area into evenly spaced grids, with each grid cell representing a block area.

At the group level, we count the inflow and outflow of agents between block areas, calculate the Origin-Destination (OD) matrix [9], and normalize it. To compare real data with simulation data, We utilize the Common Part of Commuters (CPC) to measure the structural similarity between the simulated matrix $\hat{T}$ and the ground-truth matrix T:

$$CPC(T, \hat{T}) = \frac{2\sum_{u,v}\min(T_{uv}, \hat{T}_{uv})}{\sum_{u,v}T_{uv} + \sum_{u,v}\hat{T}_{uv}}$$

A higher *CPC* value demonstrates the system's proficiency in reproducing city-wide commuting patterns and traffic distributions.

At the social domain, we calculate the income segregation index [14] for each block area. The income segregation of a place α is defined as $S_\alpha = \frac{2}{3}\sum_q |\tau_{q\alpha} - \frac{1}{4}|$, where $\tau_{q\alpha}$ is is the proportion of visitors in each income quartiles for place α. The S_α ranges from 0 to 1. A high S_α indicates that the place α is predominantly visited by a single income group, suggesting a high level of income segregation. We calculate the Pearson correlation coefficient between the simulated and real-world mean S_i across different urban regions. This assesses the framework's ability to identify spatial segregation hotspots. We measure the overall distributional consistency (JSD) of the experienced segregation index across the entire population.

C The IPL Algorithm

Algorithm 1 In-context Prototype Learning (IPL) Algorithm

Input: Set of all agent static profiles $\{s_i\}_{i=1}^N$; initial group size M; similarity threshold T.
Output: A set of agent groups $\mathbf{G} = \{g_1, \ldots, g_k\}$; a set of corresponding group descriptions $\mathbf{D} = \{d_1, \ldots, d_k\}$.
// Initialization Phase
Let $\{s_i\}_{i=1}^M$ be the profiles of the first M agents.
Send $\{s_i\}_{i=1}^M$ to an LLM with a prompt to cluster them and generate a concise textual description for each resulting cluster.
Initialize $\mathbf{G}$ and $\mathbf{D}$ with the clusters and descriptions returned by the LLM.
// Iterative Classification Phase
for each remaining agent j from $M+1$ to N **do**
  $max\_score \leftarrow -1$
  $best\_group \leftarrow$ null
  for each group description $d_k \in \mathbf{D}$ **do**
    Send s_j and d_k to an LLM with a prompt to evaluate their semantic similarity score (from 0.0 to 1.0).
    Let $score_k$ be the returned similarity score.
    if $score_k > max\_score$ **then**
      $max\_score \leftarrow score_k$
      $best\_group \leftarrow g_k$
    end if
  end for
  if $max\_score \geq T$ **then**
    Add agent j to $best\_group$.
  else
    Create a new group g_{new} with agent j.
    Generate a new description d_{new} for g_{new} using the LLM.
    Add g_{new} to $\mathbf{G}$ and d_{new} to $\mathbf{D}$.
  end if
end for
return $\mathbf{G}, \mathbf{D}$

D Image supplements

Table A1: Basic information about the dataset

Source	City	Users	Trajectory Points	Duration
[22]	Beijing	100000	297363263	Oct. 2019 - Dec. 2019
Safegraph	New york	Aggregated	760493	May 2023 - July 2023
	San Francisco	Aggregated	316732	
Foursquare	London	9409	173268	Apr. 2012 - Sept. 2013
	Paris	5809	85679	
	Sydney	1720	54170	

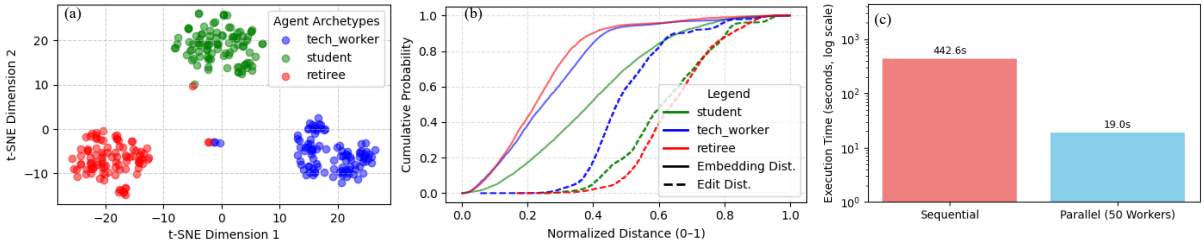

Figure A1: Empirical analysis of the scalability bottlenecks. (a) A t-SNE visualization of agent profiles reveals distinct semantic clusters. (b) Cumulative distribution plots show that agents within a cluster are semantically similar (low embedding distance) but syntactically diverse (high edit distance). (c) A time cost breakdown confirms the simulation is I/O-bound and highlights the potential of parallelization.

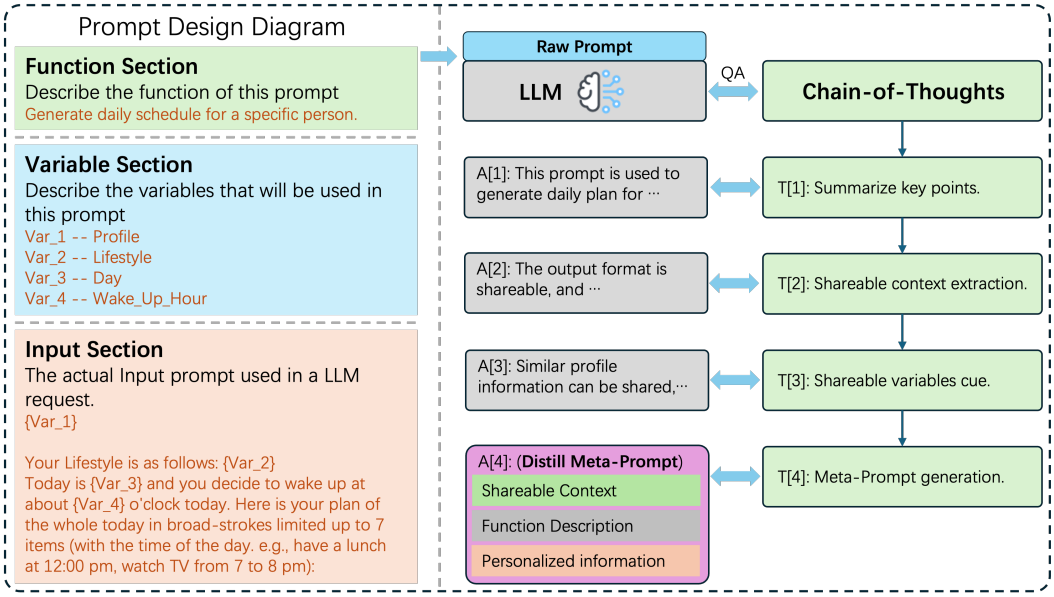

Figure A2: Distill meta-prompt generation through CoT inference.

Human Input	What places did you go today and why?	How many people did you see today? How were they?
Agent Output	About the places I visited today and the reasons: 1.Sleep at home (00:00 - 10:00): Resting to start the day. 2.Brunch (10:00 - 11:32): Enjoying a leisurely meal. 3.Outdoor Activity (12:06 - 14:45): Engaging in physical activity 4.Curry Boyz (15:43 - 17:10): Likely for a meal, perhaps trying Indian cuisine. 5.Walgreens (17:53 - 20:38): To pick up personal care items or medication. 6.Blush! Wine Bar (21:23 - 23:02, and stayed until late): Relaxing, enjoying wine, and possibly socializing. Each place was chosen for a mix of relaxation, exercise, dining, and socializing.	The people I met today: 1. When I went to Curry Boyz, I met [547] and [1032], [547] is an IT engineer, [1032] is a teacher, [547] has a high level of consumption, [1032] has a medium level of consumption. 2. When I went to Blush! Wine Bar, I met 1032 and 779, [779] is a worker, [779]'s consumption level is slightly low. The above is about the people I met today and what they were up to.

Figure A3: Illustation of the case study.