TrafficSimAgent: A Hierarchical Agent Framework for Autonomous Traffic Simulation with MCP Control

Yuwei Du[§], Jun Zhang[§], Jie Feng[§], Zhicheng Liu[†], Jian Yuan[§], Yong Li[§]

[§]Department of Electronic Engineering, BNRist, Tsinghua University, Beijing, China

[†]AMAP, Alibaba Group, Beijing, China

ABSTRACT

Traffic simulation is important for transportation optimization and policy making. While existing simulators such as SUMO and MAT-Sim offer fully-featured platforms and utilities, users without too much knowledge about these platforms often face significant challenges when conducting experiments from scratch and applying them to their daily work. To solve this challenge, we propose TrafficSimAgent, an LLM-based agent framework that serves as an expert in experiment design and decision optimization for general-purpose traffic simulation tasks. The framework facilitates execution through cross-level collaboration among expert agents: high-level expert agents comprehend natural language instructions with high flexibility, plan the overall experiment workflow, and invoke corresponding MCP-compatible tools on demand; meanwhile, low-level expert agents select optimal action plans for fundamental elements based on real-time traffic conditions. Extensive experiments across multiple scenarios show that TrafficSimAgent effectively executes simulations under various conditions and consistently produces reasonable outcomes even when user instructions are ambiguous. Besides, the carefully designed expert-level autonomous decision-driven optimization in TrafficSimAgent yields superior performance when compared with other systems and SOTA LLM based methods.

1 INTRODUCTION

Traffic simulation is of paramount importance as a critical platform for optimizing transportation infrastructure development, refining traffic management policies, and enhancing overall travel efficiency [2, 25]. Furthermore, traffic simulation plays a vital role across various interdisciplinary fields [36], such as modeling dynamic human flow in urban planning, or simulating vehicle movement patterns for environmental monitoring and management. Given this significant value, numerous traffic simulation platforms have emerged, each designed for specific formatting and modeling needs, including tools like SUMO [2], MATSim [25], CityFlow [35], and MOSS [36]. These platforms offer diverse tools to address and optimize various traffic-related challenges.

However, the effective use of these existing platforms relies heavily on expert knowledge [14, 25, 32]. This dependence creates significant barriers for many users, particularly those from interdisciplinary backgrounds, resulting in several key difficulties: preparing simulation data is challenging, intervening in the simulation process is complex, and interpreting and optimizing the simulation results is often difficult. These issues severely limit the practical application of these platforms and hinder the effective optimization of real-world problems.

In recent years, the emergence of Large Language Models, endowed with vast common sense and powerful reasoning and planning capabilities, has made it possible to build automated workflows for complex real-world tasks. This approach has seen rapid development in fields such as WebAgent [12, 38] and CodingAgent [18, 30], with notable examples like WebArena [38] and SWE-Agent [30]. This success has inspired researchers in the transportation domain, leading to several new works, including TrajAgent [9], ChatSUMO [14], and SUMO-MCP [32]. However, TrajAgent [9] focuses on modeling movement trajectories, such as mobility data generation, and does not directly support traffic simulation. While both ChatSUMO [14] and SUMO-MCP [32] support SUMO-based traffic simulation, they are limited to the automated execution of a fixed set of scenarios and parameters. Crucially, they all lack systematic automatic optimization and feedback capabilities, preventing effective iterative improvement and optimization.

To address the aforementioned challenges, we propose TrafficSimAgent, a novel LLM based multi-agent framework for autonomous traffic simulation. Specifically, TrafficSimAgent introduces three core components: first, it defines and abstracts API functions, decoupling them from the underlying traffic simulation platform and packaging them as MCP Functions to support automatic calling by the LLM Agents; second, it features a Task Understanding and Autonomous Planning module that automatically interprets vague natural language instructions, enabling the agent to freely combine underlying functional modules and break away from the fixed workflows, which significantly enhances the generality and adaptability of the automated simulation system to different scenarios; and finally, integrated with the simulation process, it includes a built-in system optimization module that supports both a full-stack automatic optimization driven entirely by LLM and a two-layer optimization logic combining the LLM's strengths with classical low-level optimization algorithms.

The main contributions of this paper are summarized as follows:
- To the best of our knowledge, we are the first to propose an autonomous traffic simulation agent capable of both automatic optimization and scenario generalization.
- We introduce a set of Abstracted API Functions, decoupled from the underlying traffic simulation platform and exposed via the
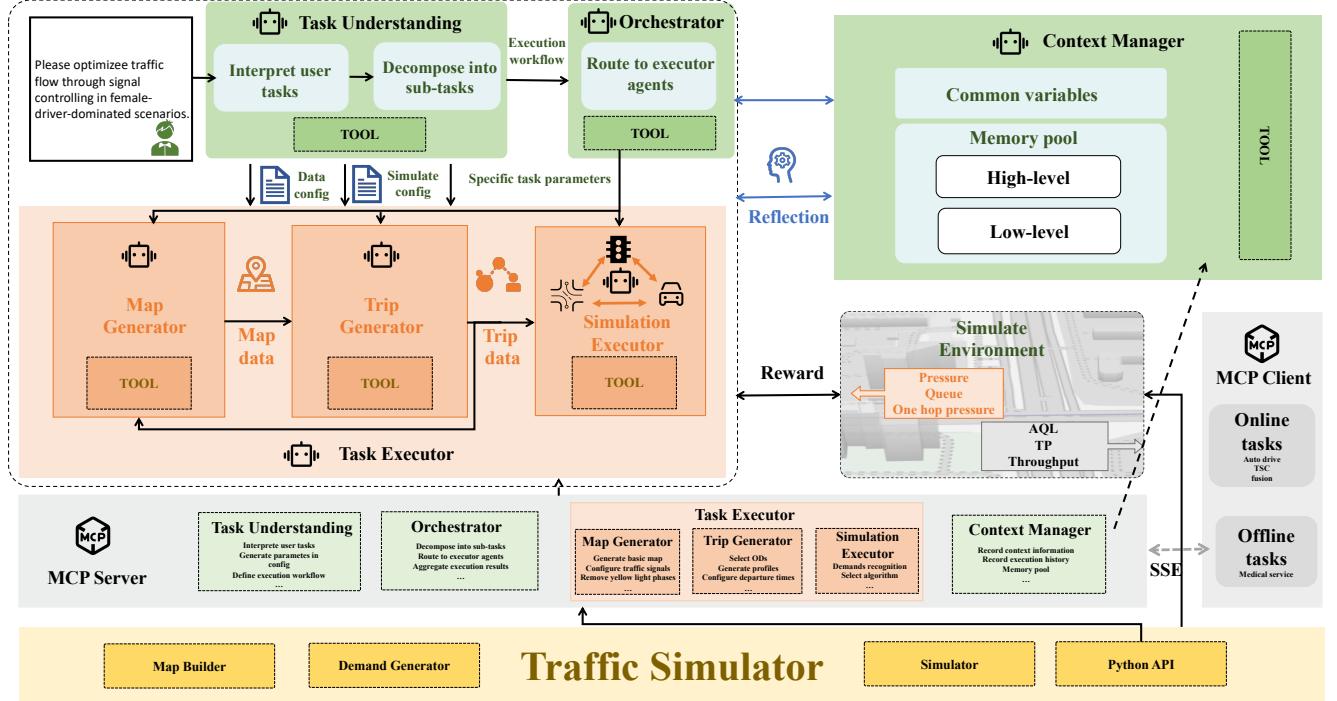
Figure 1: The whole framework of *TrafficSimAgent* .

MCP interface, establishing a foundational infrastructure for general-purpose autonomous traffic simulation.

- We design a novel Task Understanding and Autonomous Planning module that supports the interpretation of ambiguous natural language instructions and enables multi-scenario generalization, thus allowing the system to adapt to diverse environments.

- We propose an Embedded Optimization Module featuring a two-layer optimization system: one based on fully autonomous LLM-driven optimization, and another combining the LLM with classical algorithms. This enables self-optimization of the simulation and performance improvement.

- Extensive experiments demonstrate the generalization and stability of our proposed framework across various simulation scenarios. Furthermore, its automatic optimization capability enables automated, iterative performance improvement tailored to specific task requirements.

2 METHODS

TrafficSimAgent is designed as a collaborative multi-agent framework that bridges natural language instructions and traffic simulation execution. As shown in Figure1, our framework consists of four core modules working together: Task Understanding Module, Orchestrator Module, Task Executor Module, and Context Manager Module. Unlike open-ended agent frameworks like OpenManus that lack task-specific structure, TrafficSimAgent incorporates domain knowledge of transportation experiments through specialized module design, ensuring both flexibility and execution reliability.

2.1 Overall Framework

LLM agents act as experiment planners and optimizers. According to the general implementation of transportation experiments, we design a task understanding module, three task execution modules—including a map generator module, a trip generator module, and a simulation executor module—each managed by an LLM agent. To assist each agent in better understanding the specific implementation plan and enabling efficient collaboration among agents, we design an Orchestrator module that serves as a global planner and a Context Manager that functions as a shared memory pool. The workflow begins with the Task Understanding Module interpreting user instructions. The Orchestrator then decomposes the task into subtasks and routes them to appropriate executors. Each executor module organizes experiments automatically and invokes necessary tools via the MCP server. Throughout this process, the Context Manager maintains execution history and agent memories, enabling reflection and learning. The server, built on the MCP-agent framework, dynamically registers and runs only the necessary tools—such as MOSS utilities and other auxiliary utilities—via a lightweight JSON-RPC interface.

2.1.1 MCP client. **Task Understanding Module** The module understands the user's instructions, then recognizes key parameters for each executor module; detailed parameters are provided in Table1. The module also performs global planning, assigning the execution order of each module according to the user's instructions. For example, the instruction: "compare the TSC experimental results of Yizhuang during morning peak hour and Shanghai at midnight" corresponds to the following execution order: map generator →

trip generator → simulation executor → map generator → trip generator → simulation executor.

Task Executor Module The module consists of three sub-modules.

(1)Map Generator: Generate traffic network maps according to the parameter region. First, use the geocoding service of OSM (OpenStreetMap) to obtain the coordinates of the four corners of the region boundary. Then, fetch POI, AOI, street, lane, and junction data within the boundary from OSM in GeoJSON format. Finally, use GeoBuilder to convert the GeoJSON to Protocol Buffer (Protobuf) format, while also performing lane reconstruction and traffic light generation within intersections.

(2)Trip Generator: Generate trip files for vehicles and persons for simulation. First, generate user profiles and departure time curves in parallel with the distribution described in the user instructions. Then, generate the OD matrix (Origin-Destination matrix) for persons and vehicles (drivers) in the target region. Finally, set personalized driving patterns for each driver and use routing algorithms to generate travel paths based on the OD matrix. It also supports post-processing map data if required for specific tasks. For example, removing the yellow light phase from the map for TSC (Traffic Signal Control) tasks.

(3)Simulation Executor: Execute simulation tasks and choose the optimal execution strategy according to the user instructions. We categorize tasks into online simulation and offline simulation. Offline simulation involves modifying static elements, such as AOI, street, lane, and junction data. Online simulation involves modifying dynamic elements by analyzing real-time information collected during the simulation, such as adjusting vehicle driving states based on road conditions and changing traffic light phases at intersections in real time. Online simulation supports both traditional RL algorithms and LLM agent-controlled methods to collect environmental information, calculate current rewards, and determine update strategies for future states of dynamic elements.

Orchestrator Module This module functions as an agent router, decomposing complex tasks into sub-tasks and routing each sub-task to the relevant module's agent. It supports multi-agent parallel task execution, monitors the execution process, and aggregates the execution results from each module.

Context Manager Module This module consists of a global context manager and multiple individual memory managers. The global context manager oversees the execution results of each module and shares key parameter information between modules. Once the task understanding module receives the user's instructions, the context manager creates a new session to record the context history of each task executor module, including tool parameters, tool execution order, and error messages. Task executor modules can perform reflection based on this history to enhance task execution success rates. The memory managers record the decision history of each agent in the simulation executor module, adaptively updating as simulation time progresses, enabling agents to recognize and learn from the highest-reward decisions.

2.1.2 MCP Server. **Tools** The tool set is mainly organized into four groups, corresponding to the task understanding module and the three task execution modules. Each group remains inactive until the
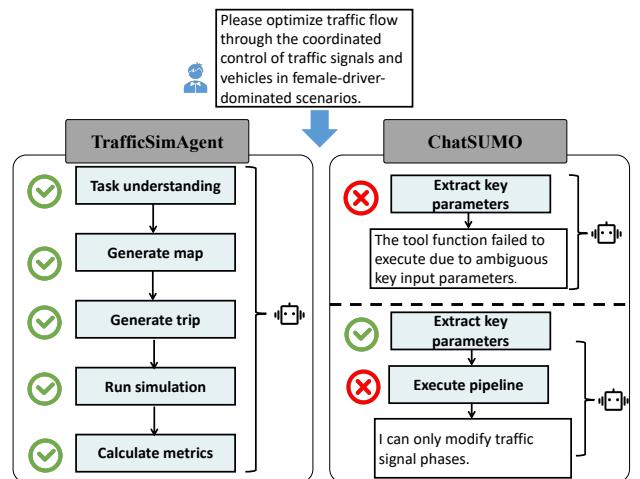

Figure 2: Case study example

corresponding module is executed. There are also some auxiliary tools in Table1 to facilitate a higher task execution success rate.

2.2 Design for Enhanced Generality

The architecture of TrafficSimAgent is specifically designed to achieve superior generality across diverse simulation scenarios, overcoming the limitations of previous systems that rely on predefined workflows and limited task support. This enhanced generality is realized through three key design innovations: robust instruction understanding, dynamic task planning, and flexible element-agent embodiment.

2.2.1 Robust Instruction Understanding through Semantic Comprehension. Unlike pattern-based approaches such as ChatSUMO that extract predetermined parameters through template matching, TrafficSimAgent 's Task Understanding Module employs deep semantic comprehension to handle ambiguous, incomplete, or context-dependent instructions. The module utilizes LLM-powered analysis to infer implicit parameters and contextual relationships, then structures this understanding into standardized configuration files for downstream processing. This approach significantly expands the range of acceptable instructions beyond predefined templates, enabling natural and flexible user interaction while maintaining structured data flow between modules.

2.2.2 Dynamic Task Planning through Hierarchical Decomposition. Traditional systems like ChatSUMO employ fixed execution workflows that limit their applicability to predetermined scenarios. TrafficSimAgent overcomes this limitation through its Orchestrator Module, which dynamically plans and adapts execution sequences based on semantic understanding of each unique instruction. We show a case in Figure2.

The dynamic planning process enables handling of complex, multi-faceted instructions that would require manual workflow specification in previous systems. For instance, the comparative

instruction "analyze traffic patterns in Shanghai between morning and evening peaks, then optimize signal timing for the worst-performing period" triggers the following adaptive workflow: (1)Map generation for Shanghai region. (2)Trip generation with morning peak profile parameters. (3)Simulation execution and metric collection. (4)Trip regeneration with evening peak profile parameters. (5)Simulation execution and metric collection. (6)Comparative analysis and bottleneck identification. (7)Targeted optimization of identified critical intersections. (8)Validation simulation and results reporting. This flexible planning capability allows TrafficSimAgent to accommodate diverse instruction types without requiring predefined workflow templates, significantly expanding its applicability across different experimental designs.

2.2.3 Diverse Task Support through Element-Agent Embodiment. The framework's generality is further enhanced by its foundational principle of element-agent embodiment, where each fundamental traffic element (e.g., an intersection, a vehicle) is represented by an LLM agent that makes autonomous decisions in real-time based on environmental information. This architecture supports a wide range of simulation tasks through emergent agent behaviors rather than hardcoded scenario implementations. For example, the fusion task requires coordinated decision-making between agents role-playing traffic signals and those role-playing vehicles. Each agent optimizes its behavior while considering the others' states and intentions, without the need for pre-programmed coordination logic.

2.3 Design for Autonomous Optimization

TrafficSimAgent 's autonomous optimization capability is achieved by employing LLM agents to act as fundamental traffic elements and make coordinated decisions based on real-time states, moving beyond merely functioning as an on/off switch for the simulation process. This optimization operates at two levels: high-level optimization strategy selection and low-level real-time collaborative optimization. The Context Manager Module enables agents to reflect on their execution records and rewards from environment, continuously improving success rates.

2.3.1 High-Level Optimization Strategy Selection. The Orchestrator Module is able to automatically select the most suitable optimization approach based on user instructions and task characteristics. For tasks requiring precise numerical optimization (e.g., "minimize average travel time"), it may invoke traditional optimization algorithms such as MaxPressure or reinforcement learning-based methods. For tasks demanding commonsense reasoning and complex trade-offs (e.g., "reduce congestion while maintaining safety"), it employs LLM-based optimization strategies. This adaptive selection mechanism ensures optimal performance across diverse scenarios.

2.3.2 Low-Level Real-Time Collaborative Optimization. During simulation execution, low-level agents achieve real-time closed-loop optimization of traffic conditions through perception-decision-action cycles. Each agent (e.g., traffic signal or vehicle) makes decisions considering not only its own state but also the cooperative status of other elements and real-time traffic conditions information. For

instance, a traffic signal agent adjusts phase timing by comprehensively analyzing current queue lengths, pressure in the neighbourhood, the intentions of approaching vehicles, and overall regional traffic density. Similarly, a vehicle agent selects optimal acceleration and speed while considering signal states ahead and the cooperative behavior of surrounding vehicles. This multi-element collaborative decision-making mechanism overcomes the limitations of traditional algorithms (e.g., MaxPressure, MPLight) that utilize only localized neighborhood information, and existing LLM-based methods (e.g., LLMLight) that inadequately consider the status of other elements

3 EXPERIMENTS

TrafficSimAgent utilizes MOSS as the simulator with an MCP-agent. Users input natural language instructions at the client side, and related tools are subsequently invoked at the server side. Variables parsed from these instructions and intermediate variables (for example, data paths and the boundary coordinates of the target region) are stored in the context manager. Instead of relying on a predefined workflow and setup, we employ an Orchester module to plan the execution sequence and assign reasonable values to parameters in the configuration file, guided by the natural language instructions. We also enable LLM agents to optimize the experimental results during the simulation process, rather than merely functioning as an on/off switch. Therefore, we primarily aim to demonstrate the versatility and autonomous optimization capability of TrafficSimAgent.

3.1 Experimental Settings

3.1.1 Baseline Models and Methods. To comprehensively evaluate the performance of TrafficSimAgent, we compare it against a range of baseline approaches, categorized into two groups for the main experiment and the optimization experiment, respectively.

Baselines for Main Experiment We select baselines that represent different paradigms in task automation and language-driven simulation:

- Domain-Specific Agent Frameworks: ChatSUMO [14], a specialized framework that integrates LLMs with the SUMO simulator for traffic simulation tasks.
- General-Purpose Agent Frameworks: OpenManus [15] and MetaGPT [11], which are state-of-the-art, multi-agent frameworks designed for general software task execution but not specifically tailored for traffic simulation.
- General Large Language Models (LLMs): GPT-5 and Gemini 2.5 Pro [6]. These powerful models are prompted to perform the simulation tasks in a zero-shot or few-shot manner, representing the capability of raw LLMs without specialized frameworks.

Baselines for Optimization Experiment To specifically assess the optimization capability of our low-level agents, we compare against established Traffic Signal Control (TSC) algorithms:

- MaxPressure [16]: A classic, model-free TSC algorithm that controls signals based on the real-time pressure (queue length) of incoming and outgoing lanes.
- MPLight [5]: A deep reinforcement learning-based method that utilizes the MaxPressure principle as part of its reward function for adaptive signal control.

Table 1: Modular architecture and functional components of TrafficSimAgent.

module name	submodule name	tools	parameters
Task Understanding	–	analyze-requirement, extract-key-parameter, validate-parameters	natural_language_input
Orchestrator	–	agent_router	–
Task Executor	Map Generator	generate-basic-map, configure-traffic-signals, preprocess-map-for-tsc	region name, green_time, yellow_time
	Trip Generator	select-origins-destinations, generate-profiles, configure-departure-times, generate-persons-vehicles, configure-personalized-driving	boundary_coordinates, age_ranges, cons_ranges, gender_distribution, education_distribution, start_step, duration_step, persons_num, vehicles_num
	Simulation Executor	demand-recognition, select-algorithm, execute-scenario, monitor-simulation-progress, extract-simulation-metrics	algorithm, scenario_name, reward_type, llm_control_interval, start_step, duration_step
Context Manager	Common Variables	create_session, get_session, export_session, import_session, record_tool_call, get_tool_call_history, get_agent_state, update_agent_state	session_id
	Memory Pool	get_agent_background, record_decision, add_background_knowledge, search_conversation, get_conversation_summary, clear_memory, health_check	session_id, max_conversation_length, max_session_memory, memory_retention_days, auto_summarize_interval, background_info_length, max_context_variables, context_summary_length

- LLMLight [13]: A state-of-the-art LLM-based TSC method that leverages large language models for decision-making at intersections.

3.1.2 Dataset Construction.
All experimental data were generated on-demand by our TrafficSimAgent via the MOSS simulation environment, including a map network and travel demand data. The data generation was directly guided by the user's natural language instructions, ensuring each scenario accurately reflects the specified conditions.

Map Network Dataset comprises the fundamental topological and geographical elements required for traffic simulation, which is generated by the Map Generator in executor module through the following procedure:

(1)Geocoding: Given a region name (e.g., "Yizhuang, Beijing"), we utilize the OpenStreetMap (OSM) geocoding service to obtain its precise geographical boundaries (i.e., the four-corner coordinates).

(2)Data Fetching & Conversion: The bounding box is used to fetch raw map data from OSM, which is then converted into a simulation-ready network using the MOSS simulator's utilities. The final map dataset comprises several core elements: lanes, roads, junctions, and AOIs (Areas of Interest).

Travel Demand Dataset defines the movement of individuals and vehicles, which is generated by the Trip Generator in executor module through the following procedure:

(1)OD Matrix Generation: An Origin-Destination (OD) matrix is first generated based on an analysis of map dataset generated by Map Generator.

(2)Individual Demand Synthesis: The aggregate OD matrix is then disaggregated into individual travel demands. This critical step is guided by the natural language instruction and involves intelligently determining several key parameters. To enhance the robustness of this process, Context Manager Module enables reflective validation, ensuring the selected parameters and scenario

configurations are contextually appropriate and consistent with the user's intent. Key parameters are as follows:

- Population Size: The total number of users to simulate.
- Origin-Destination Selection: Choosing specific start and end points (AOIs) that reflect the instructed scenario.
- Travel Mode: Assigning modes of transport (e.g., car, taxi, walk).
- User Profiles: Generating demographic profiles for each individual, including age, gender, education, and consumption level.
- Departure Time Distribution: Configuring a time distribution curve for trip departures (e.g., a peak-shaped curve for "rush hour").
- Driving Parameters: Configuring either personalized or unified driving behavior parameters (e.g., usual acceleration, average speed) for each vehicle.

(3) Simulation Integration: The generated individual trips are finally integrated with the road network for execution in the simulator. The key parameters used in the travel demand generation are summarized in Table 1.
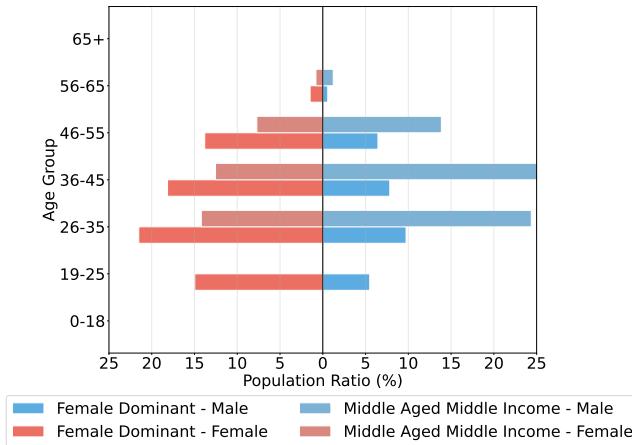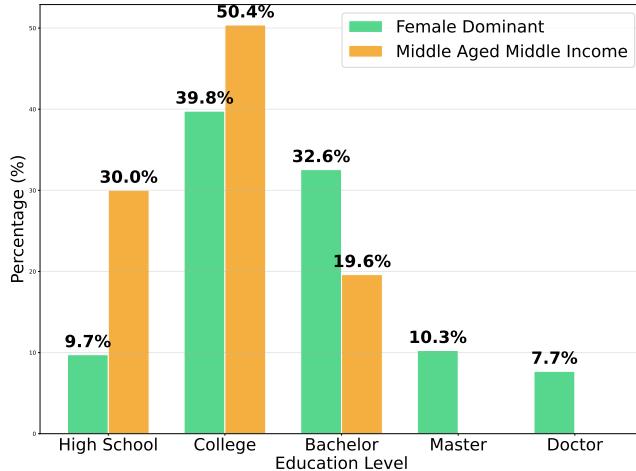
3.1.3 Metrics.
To comprehensively evaluate the performance of all methods, we employ a set of metrics from traffic simulation, categorized into efficiency and environmental impact dimensions. Efficiency is measured by traffic volume (TV), cumulative throughput, travel time (ETA, ATT-finished), queue length (AQL), and ETA reduction (ΔETA); environmental impact is gauged by total and average carbon emissions (Carbon, ACE). Formal definitions are provided in Appendix A.

3.2 Generalization
To demonstrate the ability of TrafficSimAgent to handle various types of natural language instructions, we designed a natural language instruction benchmark covering diverse scenarios across

Table 2: Performance comparison of simulation methods. Best and second-best results are highlighted in bold and <u>underline</u>, respectively.

Model	auto drive			fusion			tsc			medical service			MRR↑
	TV↑	AQL↓	TP↑	AQL↓	ACE↓	TP↑	AQL↓	ACE↓	TP↑	serve_rate↑	ATT↓	score↑	
ChatSUMO	-	-	-	-	-	-	**18.91**	0.59	673	-	-	-	0.246
gpt-5	<u>408</u>	17.04	1116	<u>18.63</u>	1.44	1136	30.43	0.30	1253	98%	3064.29	-0.233	0.336
gemini2.5-pro	401	<u>17.11</u>	1121	25.95	0.40	1290	22.31	0.69	1255	98.5%	1472.81	0.248	0.285
MetaGPT	94	18.29	677	**18.34**	0.75	677	<u>19.86</u>	0.59	673	91.0%	3665.58	-0.463	0.321
OpenManus	125	74.90	**2113**	74.90	<u>0.32</u>	**2113**	74.95	**0.25**	**2110**	99.5%	1283.82	0.311	<u>0.651</u>
TrafficSimAgent	**420**	**16.81**	<u>1308</u>	24.34	<u>0.29</u>	<u>1417</u>	29.10	<u>0.29</u>	<u>1337</u>	**99.5%**	**1283.82**	**0.311**	**0.715**

Figure 3: Gender and age distribution of user data generated by TrafficSimAgent for different user groups.

Figure 4: Education distribution of user data generated by TrafficSimAgent for different user groups.

different simulation tasks, categorized into "specified" and "ambiguous" instructions. The model was tasked with two ambiguous instruction to simulate driving behaviors of two distinct demographic groups: middle-aged, middle-income drivers with median education levels(Middle Aged Middle Income), and female-dominated driving populations(Female Dominant). The generated results are presented in Figure 3 and Fig 4, showing distributions that align broadly with the specified instructions, illustrating that it genuinely understands ambiguous instructions instead of mechanically parsing parameters from them.

Furthermore, we sought to evaluate the general capability of TrafficSimAgent on both online and offline simulation tasks. We selected medical service selection as the offline simulation task, and auto_drive, TSC (Traffic Signal Control), and fusion as the online simulation tasks. We compared the results of TrafficSimAgent against a general model, a general agent framework, and a domain-specific agent framework. Table 2 presents the comparison results.

The results indicate that ChatSUMO, the domain-specific model, can only handle the TSC task. GPT-5 and Gemini-pro, the general models, show limited performance in either congestion mitigation (high AQL) or economic and environmental efficiency (high carbon emission). MetaGPT, as a general agentic framework, adopts an overly conservative strategy (low TP, low AQL), whereas Open-Manusor adopts the opposite approach (high TP, high AQL). In contrast,TrafficSimAgent strikes a balance among TP, AQL, and carbon emission, enabling all vehicles to complete their trips as quickly and smoothly as possible without excessive congestion, thereby avoiding carbon emissions caused by inefficient acceleration near the speed limit.

3.3 Autonomous Optimization

We evaluated the traffic optimization results by comparing TrafficSimAgent with traditional TSC algorithms (MaxPressure, MPLight) and LLMLight, a state-of-the-art LLM-based TSC algorithm. Experiments were performed using the aforementioned dataset. The comparison results are shown in Table 3. TrafficSimAgent, which employs LLM agents for macro-level traffic condition optimization by controlling individual traffic elements in consideration of current traffic conditions and the status of other elements, proves to be the optimal approach. Specifically, the fusion algorithm performs best by enabling cooperative control between traffic lights and vehicles. LLMLight fails to adequately consider the information from other elements, while traditional algorithms only utilize neighborhood information, leading to a suboptimal overall optimization outcome.

Table 3: Performance comparison of traffic condition optimization methods. Best and second-best results are highlighted in bold and <u>underline</u>, respectively.

|	Opt Methods	ATT-f↓	TV↑	AQL↓	ΔETA↑	ACE↓	ETA↓	TCE↓	TP↑	**MRR↑**
Others (TSC)	MPLight	165.15	143	17.44	-161.98	<u>0.84</u>	15487651.37	<u>440.92</u>	526	0.360
	MaxPressure	200.27	215	18.9142	-164.79	0.87	**16253681.12**	476.05	551	0.257
	LLMLight	312.92	**1027**	**13.486**	-162.53	1.44	<u>17836227.97</u>	853.84	595	0.405
Ours (CollabOpt)	auto-drive	355.09	<u>420</u>	<u>16.81</u>	-142.35	1.22	33166900.16	1355.3	1308	0.290
	fusion	<u>134.68</u>	120	24.34	**-125.73**	0.29	39663544	**420.67**	**1417**	**0.564**
	TSC	**125.42**	105	29.10	<u>-139.53</u>	**0.29**	39634600.37	387.76	<u>1337</u>	<u>0.557</u>
no-control	–	354.55	420	25.07	-156.75	1.13	32943916.96	1249.22	1107	0.244
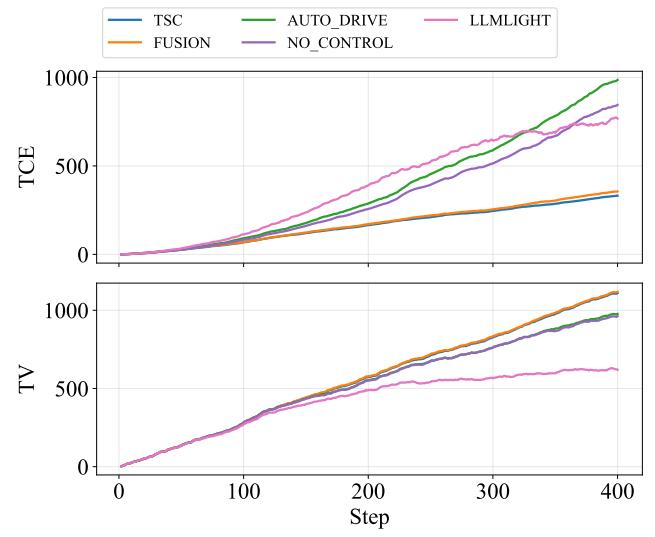
3.4 Analysis on Agent Reasoning and Model Scaling

To analyze the advantages of our framework over traditional LLM-based TSC implementations like LLMLight, which typically provides junction phase information to traffic signal agents at each step, we compare the evolution of traffic throughput and carbon emission across simulation steps, as shown in Figure 5. The experimental results reveal that our method's superiority becomes increasingly pronounced as simulation progresses. This emergent advantage stems from a fundamental difference in decision-making paradigms. While conventional approaches like LLMLight rely on fine-grained phase analysis (providing per-phase queue lengths for each junction), our low-level agents operate under a reinforcement learning-inspired philosophy that maximizes a composite reward function incorporating pressure differences between incoming and outgoing lanes, total queued vehicles, and neighboring junction influences.

This reward-driven approach creates a critical shift from short-term reactive decisions to long-term strategic optimization. The phase-specific information used by traditional methods often leads to greedy decisions that minimize immediate queue lengths but disrupt the balance between intersection pressure and overall traffic throughput. In contrast, our agents leverage historical trend analysis through the scratchpad mechanism (e.g., strategy: try-better-action) to learn from long-term outcomes rather than reacting to instantaneous snapshots.

Our goal-oriented reward signal combined with memory-enabled strategic decision-making drives more effective long-term optimization. Besides, Historical State maintains strategy effectiveness through action-reward memory; Other Junctions enables coordinated multi-agent optimization; and the Scratchpad provides essential trend analysis for adaptive decision-making. In conclusion, our framework's growing advantage over traditional phase-analysis approaches underscores the effectiveness of replacing detailed perceptual data with goal-driven, history-aware decision processes for complex dynamic optimization in traffic signal control.

Our experiments with varying model sizes in Table 4 demonstrate a clear positive correlation between parameter count and performance. The Qwen3-235B model achieves overall superior results, while smaller models show progressively reduced capability. This confirms that TrafficSimAgent 's performance scales with the underlying model's capacity.

Figure 5: Metrics variation with simulation steps of different optimization methods.

4 RELATED WORK

4.1 Traffic Simulation

Traffic simulation has long been established as a cornerstone of urban computing and intelligent transportation systems research[22, 24, 31]. High-fidelity, open-source simulators such as SUMO [2], MOSS [36], MATSim [25] and CityFlow [35] provide researchers with powerful platforms for modeling complex traffic dynamics, evaluating traffic management policies, and testing autonomous driving algorithms. There are also traffic simulator for auto-drive, such as LCSim [37] and CARLA [8]. These tools offer extensive libraries for network modeling, vehicle movement, and traffic light control. However, conducting simulation experiment remains a non-trivial task that often poses significant challenges for users. For example, many autonomous driving-related studies based on SUMO are designed to perform each simulation step manually, such as SMDT [27], BlaFT [17], HPTSim [19], DLIO [26], ASC-HBMP [39], and RoboCar [23]. Moreover, the very process they manually execute is inherently complex, typically involving multiple intricate steps: acquiring and preprocessing map data (e.g.,

Table 4: Performance comparison of different models. Best and second-best results are highlighted in bold and <u>underline</u>, respectively.

Model	ATT-f↓	TV↑	AQL↓	ΔETA↑	ACE↓	ETA↓	TCE↓	TP↑	SR↑	MRR↑
mistral7b-v3	365.25	**215**	**12.25**	<u>-146.26</u>	2.27	37595153.88	2869.91	<u>1266</u>	35%	0.344
llama3.1-8b	<u>352.85</u>	<u>316</u>	<u>14.74</u>	-152.77	1.90	35256368.16	2248.59	1186	47%	0.287
qwen3-14b	355.28	391	16.51	-155.29	**1.11**	33556686.51	**1256.77**	1128	49%	<u>0.424</u>
deepseek-v3	**350.02**	403	16.78	-156.00	1.32	<u>33383975.97</u>	1480.04	1122	62%	0.393
llama3-70b	370.33	335	15.56	-150.59	1.62	35116956.06	1918.44	1181	<u>65%</u>	0.287
qwen3-235b	355.09	420	16.81	**-142.35**	<u>1.22</u>	**33166900.16**	<u>1355.3</u>	**1308**	**78%**	**0.611**

from OpenStreetMap), defining traffic demand (e.g., generating trip files and routes), configuring simulation parameters, and finally, executing and monitoring the simulation. This workflow requires considerable domain expertise and is largely manual, making it time-consuming, error-prone, and difficult to reproduce. Furthermore, while these simulators are capable of modeling various scenarios, they offer limited native support for automatic optimization.

4.2 LLM-Driven Simulation Automation

Recent works have begun leveraging large-language models (LLMs) within agentic frameworks to automate complex tasks through natural language instructions, demonstrating significant potential in lowering the barrier to entry for non-expert users. The emergence of autonomous AI agents has showcased LLMs' capability in task decomposition and sequential execution, while specialized systems such as ChemCrow [4] have demonstrated expert-level automation in scientific domains like organic synthesis and drug discovery. In cybersecurity, AutoPentest [10] exemplifies how LLM agents can autonomously conduct penetration testing and vulnerability assessment. In trajectory modeling, TrajAgent [9] unifies various trajectory modeling task through a collaborative agentic framework. Particularly impressive is Coscientist [3], which autonomously designs, plans, and executes real-world chemical experiments, successfully completing complex palladium-catalyzed cross-couplings. Recently there are also many works on general agent framework, such as AutoGen [29] and ChatDev [18].

This trend is particularly evident in traffic simulation domains, where LLM-powered agents can interpret high-level commands and execute corresponding simulator actions. For instance, Chat-SUMO [14] employs LLMs as a natural language interface to the SUMO simulator, allowing users to issue intuitive commands (e.g., "I want to see traffic in Albany...") that are automatically translated into executable simulator configurations. Similarly, Open-TI [7] proposes an agentic workflow where users provide natural language instructions through an LLM interface, which then autonomously starts and manages SUMO simulations in the background. These approaches exemplify the broader movement toward LLM-based automation systems that transform abstract user requirements into precise technical operations.

There are also works propose LLM-assisted frameworks to solve traffic signal control (TSC) problems, integrating SUMO for traffic simulation and algorithm validation, such as CityLight [34], LLMAATSC [20], LLMLight [13], CoLLMLight [33], LLM-assisted

light [28], and LLMDUTSC [21]. A more recent and powerful paradigm is the use of Model Context Protocol (MCP) [1]. Frameworks like SUMO-MCP [32] begin to leverage this concept by wrapping SUMO's functionalities into MCP-compatible tools. However, these works are constrained by several key limitations. Firstly, their capabilities are often bounded by pre-defined, rigid workflows, lacking the flexibility to dynamically plan and adapt a comprehensive experimental procedure from scratch. Secondly, their generality is limited: they are often tightly coupled with a narrow set of tasks (e.g., TSC), making it difficult to extend them to a broader range of scenarios. Most critically, they lack a robust mechanism for outcome optimization or fails to enhance global performance, such as minimizing overall travel time, congestion or carbon emission.

5 CONCLUSION

In this paper, we introduced TrafficSimAgent , a novel LLM-based multi-agent framework designed to overcome the generality and optimization bottlenecks in traditional traffic simulation workflows. By leveraging a hierarchical architecture of collaborative agents and the carefully designed model context protocol for traffic simulation, TrafficSimAgent translates natural language instructions into executable simulation plans and enables real-time, intelligent optimization of traffic conditions.

Our work makes several key contributions. First, we demonstrate that TrafficSimAgent possesses enhanced generality, capable of handling both specified and ambiguous instructions across diverse online and offline simulation scenarios. Second, we showcase its autonomous optimization capability, where low-level agents, embodying traffic elements, make coordinated decisions that outperform traditional and contemporary LLM-based TSC algorithms. Our analysis on agent reasoning components further shows that good performance stems from a combination of a goal-oriented reward signal and a memory-driven strategy.

Despite its promising results, the current framework has some limitations. The performance and token consumption are inherently tied to the capabilities and scale of the underlying general-purpose LLMs. For future work, we plan to investigate task-specific fine-tuning of more compact language models to reduce computational and token overhead while maintaining performance. Furthermore, we will explore to extend the framework to support multi-modal instructions and integrating it with a broader range of urban simulation platforms.

REFERENCES

[1] Anthropic. 2024. Model Context Protocol. https://www.anthropic.com/news/model-context-protocol Accessed: 2024-11-20.

[2] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. 2011. SUMO–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind.

[3] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature* 624, 7992 (2023), 570–578.

[4] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376* (2023).

[5] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. 2020. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 3414–3421.

[6] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261* (2025).

[7] Longchao Da, Kuanru Liou, Tiejin Chen, Xuesong Zhou, Xiangyong Luo, Yezhou Yang, and Hua Wei. 2024. Open-ti: Open traffic intelligence with augmented language model. *International Journal of Machine Learning and Cybernetics* 15, 10 (2024), 4761–4786.

[8] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

[9] Yuwei Du, Jie Feng, Jie Zhao, Jian Yuan, and Yong Li. 2024. TrajAgent: An LLM-based Agent Framework for Automated Trajectory Modeling via Collaboration of Large and Small Models. *arXiv preprint arXiv:2410.20445* (2024).

[10] Julius Henke. 2025. AutoPentest: Enhancing Vulnerability Management With Autonomous LLM Agents. *arXiv preprint arXiv:2505.10321* (2025).

[11] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. International Conference on Learning Representations, ICLR.

[12] Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5295–5306.

[13] Siqi Lai, Zhao Xu, Weijia Zhang, Hao Liu, and Hui Xiong. 2025. Llmlight: Large language models as traffic signal control agents. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 2335–2346.

[14] Shuyang Li, Talha Azfar, and Ruimin Ke. 2024. Chatsumo: Large language model for automating traffic scenario generation in simulation of urban mobility. *IEEE Transactions on Intelligent Vehicles* (2024).

[15] Xinbin Liang, Jinyu Xiang, Zhaoyang Yu, Jiayi Zhang, Sirui Hong, Sheng Fan, and Xiao Tang. 2025. OpenManus: An open-source framework for building general AI agents. https://doi.org/10.5281/zenodo.15186407

[16] Hao Liu, Vikash V Gayah, and Michael W Levin. 2024. A max pressure algorithm for traffic signals considering pedestrian queues. *Transportation Research Part C: Emerging Technologies* 169 (2024), 104865.

[17] Sanggu Park, Edward Andert, and Aviral Shrivastava. 2023. Blame-free motion planning in hybrid traffic. *IEEE Transactions on Intelligent Vehicles* 9, 1 (2023), 259–268.

[18] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924* (2023).

[19] Yunyang Shi, Ziyuan Gu, Anfeng Jiang, Dinghao Zhou, and Zhiyuan Liu. 2024. Hptsim: A highway parallel traffic simulation framework for mixed concurrency scenarios. *IEEE Intelligent Transportation Systems Magazine* (2024).

[20] Yiqing Tang, Xingyuan Dai, and Yisheng Lv. 2024. Large language model-assisted arterial traffic signal control. *IEEE Journal of Radio Frequency Identification* 8 (2024), 322–326.

[21] Yiqing Tang, Xingyuan Dai, Chen Zhao, Qi Cheng, and Yisheng Lv. 2024. Large language model-driven urban traffic signal control. In *2024 Australian & New Zealand Control Conference (ANZCC)*. IEEE, 67–71.

[22] Ye Tao, Ehsan Javanmardi, Jin Nakazato, Manabu Tsukada, and Hiroshi Esaki. 2023. Flowsim: A modular simulation platform for microscopic behavior analysis of City-Scale connected autonomous vehicles. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2186–2193.

[23] Mehdi Testouri, Gamal Elghazaly, and Raphael Frank. 2025. Robocar: A rapidly deployable open source platform for autonomous driving research. *IEEE Intelligent Transportation Systems Magazine* (2025).

[24] Elise Van der Pol and Frans A Oliehoek. 2016. Coordinated deep reinforcement learners for traffic light control. *Proceedings of learning, inference and control of multi-agent systems (at NIPS 2016)* 8 (2016), 21–38.

[25] Kay W Axhausen, Andreas Horni, and Kai Nagel. 2016. *The multi-agent transport simulation MATSim*. Ubiquity Press.

[26] Bowen Wang, Xinle Gong, Peiyuan Lyu, and Sheng Liang. 2024. Iterative learning-based cooperative motion planning and decision-making for connected and autonomous vehicles coordination at on-ramps. *IEEE Transactions on Intelligent Transportation Systems* 25, 7 (2024), 8105–8120.

[27] Kui Wang, Zongdian Li, Kazuma Nonomura, Tao Yu, Kei Sakaguchi, Omar Hashash, and Walid Saad. 2024. Smart mobility digital twin based automated vehicle navigation system: A proof of concept. *IEEE Transactions on Intelligent Vehicles* 9, 3 (2024), 4348–4361.

[28] Maonan Wang, Aoyu Pang, Yuheng Kan, Man-On Pun, Chung Shue Chen, and Bo Huang. 2024. LLM-assisted light: Leveraging large language model capabilities for human-mimetic traffic signal control in complex urban environments. *arXiv preprint arXiv:2403.08337* (2024).

[29] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

[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] Mobin Yazdani, Majid Sarvi, Saeed Asadi Bagloee, Neema Nassir, Jeff Price, and Hossein Parineh. 2023. Intelligent vehicle pedestrian light (IVPL): A deep reinforcement learning approach for traffic signal control. *Transportation research part C: emerging technologies* 149 (2023), 103991.

[32] Chenglong Ye, Gang Xiong, Junyou Shang, Xingyuan Dai, Xiaoyan Gong, and Yisheng Lv. 2025. SUMO-MCP: Leveraging the Model Context Protocol for Autonomous Traffic Simulation and Optimization. *arXiv preprint arXiv:2506.03548* (2025).

[33] Zirui Yuan, Siqi Lai, and Hao Liu. 2025. Collmlight: Cooperative large language model agents for network-wide traffic signal control. *arXiv preprint arXiv:2503.11739* (2025).

[34] Jinwei Zeng, Chao Yu, Xinyi Yang, Wenxuan Ao, Jian Yuan, Yong Li, Yu Wang, and Huazhong Yang. 2024. Citylight: A universal model towards real-world city-scale traffic signal control coordination. *arXiv preprint arXiv:2406.02126* (2024).

[35] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*. 3620–3624.

[36] Jun Zhang, Wenxuan Ao, Junbo Yan, Can Rong, Depeng Jin, Wei Wu, and Yong Li. 2024. Moss: A large-scale open microscopic traffic simulation system. *arXiv preprint arXiv:2405.12520* (2024).

[37] Yuheng Zhang, Tianjian Ouyang, Fudan Yu, Lei Qiao, Wei Wu, Jingtao Ding, Jian Yuan, and Yong Li. 2024. Lcsim: A large-scale controllable traffic simulator. *arXiv preprint arXiv:2406.19781* (2024).

[38] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854* (2023).

[39] Ziqi Zong, Jiamin Shi, Runsheng Wang, Shitao Chen, and Nanning Zheng. 2023. Human-like decision making and planning for autonomous driving with reinforcement learning. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 3922–3929.

A APPENDIX

A.1 Metrics

A.2 Optimization results across different scenarios

Table 5: Evaluation Metrics Used in Experiments

Category	Metric	Abbrev.	Definition
Efficiency Metrics	Traffic Volume	TV	Instantaneous count of all vehicles actively traveling in the network at a given simulation step. Reflects real-time traffic load and network occupancy.
	Cumulative Throughput	TP	Total number of vehicles that successfully complete their journeys during the entire simulation. Reflects the network's overall service capacity and operational efficiency.
	Estimated Time of Arrival	ETA	Total estimated time required for all departed vehicles to complete their trips under current traffic conditions. Lower values indicate higher system-wide traffic efficiency and reduced delay.
	Avg. Travel Time (Finished Vehicles)	ATT-finished	Average time taken by vehicles that successfully complete their trips. Lower values indicate higher individual travel efficiency.
	Average Queue Length	AQL	Average number of vehicles waiting in queues at intersections. Lower values indicate less congestion and smoother traffic flow.
	ETA Reduction	ΔETA	Difference between actual travel time and ETA under free-flow conditions. A larger negative value indicates more efficient routing and control.
Environmental Impact	Total Carbon Emission	Carbon	Total CO_2 emissions (in grams) produced by all vehicles during the simulation, calculated based on vehicle attributes (e.g., engine type) and dynamic driving behaviors (e.g., acceleration, speed). Lower values indicate a more environmentally friendly strategy system-wide.
	Average Carbon Emission per Vehicle	ACE	Average CO_2 emissions per vehicle (in grams). Lower values indicate a more environmentally friendly strategy on a per-vehicle basis.

Table 6: Performance comparison of optimization methods across different scenarios.

Scenario	Sub-scenario	Optimization	ATT-f ($\downarrow$)	TV ($\uparrow$)	ΔETA ($\uparrow$)	ACE ($\downarrow$)	ETA ($\downarrow$)	TCE ($\downarrow$)	TP ($\uparrow$)
TSC	evening_peak	With	91.62	58	-146.86	0.24	62446902.90	243.49	1006
		Without	383.02	428	-175.99	1.10	41402628.51	733.87	665
	morning_peak	With	133.71	68	-147.26	0.22	32088206.27	283.07	1231
		Without	385.32	533	-176.61	1.15	21344843.20	936.34	813
Auto-drive	female_dominant	With	358.92	871	-173.34	1.18	59829552.92	2106.02	1791
		Without	351.19	914	-1175.10	1.07	58675697.80	1885.28	1756
	middle_aged middle_income	With	382.19	932	-161.34	1.67	51704583.14	2889.28	1735
		Without	375.54	979	-162.89	1.51	50560809.29	2561.97	1696

Table 7: Performance comparison of medical service optimization strategies

Scenario	Strategy	Optimization	SR (%)	ATT (s)	Score
Medical Service	Mass-benefit	With	99.0	1735.29	0.1725
		Without	92.5	3526.22	-0.4104
Pediatric Service	–	With	100.0	2311.29	0.0067
		Without	97.0	4251.27	-0.5963